Provably Secure Cryptography: State of the Art and Industrial Applications

Pascal Paillier

Gemplus/R&D/ARSC/STD/Advanced Cryptographic Services

French-Japanese Joint Symposium on Computer Security

Outline

What is provable security?

Security Proofs for Signatures

Security Proofs for Encryption

Designing Cryptosystems

Proof Techniques

Present and Future Trends

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
-
-

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
-
-

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations)
-

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
-

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
- ...

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
- ...

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

GEMPLUS

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
- ...

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

GEMPLUS

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
- ...

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

GEMPLUS

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
- ...

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

Focus on Provable Security

Our ultimate goal:

- Providing evidence that a given cryptographic protocol is secure
- Find new ways of building secure protocols

Cryptographic protocols contain basic ingredients

- Asymmetric encryption schemes (and variations),
- Signature schemes (and variations),
- ...

So the first thing to do is trying to prove the security of these two primitives.

But what does it mean to be secure?

GEMPLUS

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable. . . (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable... (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

GEMPLUS

How Can One Prove Security?

Once a cryptosystem is described, how can we prove its security?

- By trying to mount an attack
  - Attack found $\Rrightarrow$ SYSTEM INSECURE!
  - Attack not found $\Rrightarrow$ NOTHING CAN BE SAID
- By proving that no attack exists under some assumptions
  - Public verifiability of the proof
  - Attack found $\Rrightarrow$ FALSE ASSUMPTION

When a security proof is provided, no one should be able to highlight a system defect. But the assumption has to be reasonnable. . . (e.g. the Ko-Lee assumption over Braid groups was recently proven wrong).

Provable Security is Desired

Efficient proven secure schemes have been discovered

Sign. PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...

Enc. RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

Provably secure schemes are adopted in standards

Standard bodies ask for security proofs along with submissions

Provable Security is Desired

Efficient proven secure schemes have been discovered

- Sign. PSS(-R)-RSA, GHR, Cramer-Shoup, EDL. . .
- Enc. RSA-OAEP, Cramer-Shoup, . . .

There exist generic conversions to create more of them

- Sign. Fiat-Shamir heuristic applied to ZKPK
- OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, . . .

Provably secure schemes are adopted in standards

Standard bodies ask for security proofs along with submissions

GEMPLUS

Provable Security is Desired

Efficient proven secure schemes have been discovered

- **Sign.** PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...
- **Enc.** RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

- **Sign.** Fiat-Shamir heuristic applied to ZKPK
- **Enc.** OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

Standard bodies ask for security proofs along with submissions

Provable Security is Desired

Efficient proven secure schemes have been discovered

- **Sign.** PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...
- **Enc.** RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

- **Sign.** Fiat-Shamir heuristic applied to ZKPK
- **Enc.** OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

Standard bodies ask for security proofs along with submissions

Provable Security is Desired

Efficient proven secure schemes have been discovered

- **Sign.** PSS(-R)-RSA, GHR, Cramer-Shoup, EDL…
- **Enc.** RSA-OAEP, Cramer-Shoup, …

There exist generic conversions to create more of them

- **Sign.** Fiat-Shamir heuristic applied to ZKPK
- **Enc.** OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, …

Provably secure schemes are adopted in standards

- **Sign.** PSS in IEEE P1363a and PKCS#1 v2.1
- RSA-OAEP in PKCS#1 v2.0, P1363a
- DHIES in ANSI X9.63, P1363a

Standard bodies ask for security proofs along with submissions

GEMPLUS

Provable Security is Desired

Efficient proven secure schemes have been discovered

- **Sign.** PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...
- **Enc.** RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

- **Sign.** Fiat-Shamir heuristic applied to ZKPK
- **Enc.** OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

- **Sign.** PSS in IEEE P1363a and PKCS#1 v2.1.
- **Enc.** RSA-OAEP in PKCS#1 v2.0, P1363a
  DHIES in ANSI X9.63, P1363a.

Standard bodies ask for security proofs along with submissions

Provable Security is Desired

Efficient proven secure schemes have been discovered

Sign. PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...

Enc. RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

Sign. Fiat-Shamir heuristic applied to ZKPK

Enc. OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

Sign. PSS in IEEE P1363a and PKCS#1 v2.1.

Enc. RSA-OAEP in PKCS#1 v2.0, P1363a

DHIES in ANSI X9.63, P1363a.

Standard bodies ask for security proofs along with submissions

GEMPLUS

Provable Security is Desired

Efficient proven secure schemes have been discovered

- Sign. PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...
- Enc. RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

- Sign. Fiat-Shamir heuristic applied to ZKPK
- Enc. OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

- Sign. PSS in IEEE P1363a and PKCS#1 v2.1.
- Enc. RSA-OAEP in PKCS#1 v2.0, P1363a
  DHIES in ANSI X9.63, P1363a.

Standard bodies ask for security proofs along with submissions

Provable Security is Desired

Efficient proven secure schemes have been discovered

Sign. PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...

Enc. RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

Sign. Fiat-Shamir heuristic applied to ZKPK

Enc. OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

Sign. PSS in IEEE P1363a and PKCS#1 v2.1.

Enc. RSA-OAEP in PKCS#1 v2.0, P1363a
DHIES in ANSI X9.63, P1363a.

Standard bodies ask for security proofs along with submissions

GEMPLUS

Provable Security is Desired

Efficient proven secure schemes have been discovered

Sign. PSS(-R)-RSA, GHR, Cramer-Shoup, EDL...

Enc. RSA-OAEP, Cramer-Shoup, ...

There exist generic conversions to create more of them

Sign. Fiat-Shamir heuristic applied to ZKPK

Enc. OAEP(+/++), Fujisaki-Okamoto, REACT, GEM-I, GEM-II, ...

Provably secure schemes are adopted in standards

Sign. PSS in IEEE P1363a and PKCS#1 v2.1.

Enc. RSA-OAEP in PKCS#1 v2.0, P1363a
DHIES in ANSI X9.63, P1363a.

Standard bodies ask for security proofs along with submissions

GEMPLUS

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

GEMPLUS

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

- which systems to use
- how different cryptosystems compare

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

- which systems to use
- how different cryptosystems compare

GEMPLUS

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

- which systems to use
- how different cryptosystems compare

GEMPLUS

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

- which systems to use
- how different cryptosystems compare

GEMPLUS

Provable Security is Desired (Cont'd)

Provably secure schemes are found in present systems

Sign. RSA-PSS

Enc. RSA-OAEP

These are to be widely deployed, but there may be others in near future.

Provably secure schemes in upcoming systems

This is no longer just theory. Product developers, security architects and users want to know

- which systems to use
- how different cryptosystems compare

GEMPLUS

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

GEMPLUS

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

How to Get a Security Proof?

To get a security proof, one needs to

1. Describe a cryptosystem and its operational modes,
2. Formally define a **security notion** to achieve,
3. Make precise computational assumptions,
4. Exhibit a **reduction** between an algorithm which breaks the security notion and an algorithm that breaks the assumptions.

Reduction

to prove

$$P_1 \Leftarrow P_2$$

i.e. that problem P_1 is reducible to problem P_2, one shows an algorithm with polynomial resources that solves P_1 with access to an oracle that solves P_2.

GEMPLUS

Digital Signatures

- Signer Alice generates a public/private key pair (pk, sk) by running a probabilistic key generation algorithm $G(|pk|)$, $|pk|$ being the security parameter. Alice publishes pk.
- Whenever Alice wishes to sign a digital document $m \in \{0, 1\}^*$, she computes the signature $s = S(sk, m)$ where S is the (possibly probabilistic) signing algorithm. She outputs s and maybe also m.
- Knowing m and s (and Alice's public key pk), Bob can verify that s is a signature of m output by Alice by running the verification algorithm $V(pk, m, s)$ returning 1 if $s = S(sk, m)$ or 0 otherwise.

The cryptographic system given by the triple (G, S, V) and their domains is called a signature scheme.

Digital Signatures

- Signer Alice generates a public/private key pair (pk, sk) by running a probabilistic key generation algorithm $G(|pk|)$, $|pk|$ being the security parameter. Alice publishes pk.
- Whenever Alice wishes to sign a digital document $m \in \{0,1\}^*$, she computes the signature $s = S(sk, m)$ where S is the (possibly probabilistic) signing algorithm. She outputs s and maybe also m.
- Knowing m and s (and Alice's public key pk), Bob can verify that s is a signature of m output by Alice by running the verification algorithm $V(pk, m, s)$ returning 1 if $s = S(sk, m)$ or 0 otherwise.

The cryptographic system given by the triple (G, S, V) and their domains is called a signature scheme.

Digital Signatures

- Signer Alice generates a public/private key pair (pk, sk) by running a probabilistic key generation algorithm $G(|pk|)$, $|pk|$ being the security parameter. Alice publishes pk.
- Whenever Alice wishes to sign a digital document $m \in \{0,1\}^*$, she computes the signature $s = S(sk, m)$ where S is the (possibly probabilistic) signing algorithm. She outputs s and maybe also m.
- Knowing m and s (and Alice's public key pk), Bob can verify that s is a signature of m output by Alice by running the verification algorithm $V(pk, m, s)$ returning 1 if $s = S(sk, m)$ or 0 otherwise.

The cryptographic system given by the triple (G, S, V) and their domains is called a signature scheme.

Digital Signatures

- Signer Alice generates a public/private key pair (pk, sk) by running a probabilistic key generation algorithm $G(|pk|)$, $|pk|$ being the security parameter. Alice publishes pk.
- Whenever Alice wishes to sign a digital document $m \in \{0,1\}^*$, she computes the signature $s = S(sk, m)$ where S is the (possibly probabilistic) signing algorithm. She outputs s and maybe also m.
- Knowing m and s (and Alice's public key pk), Bob can verify that s is a signature of m output by Alice by running the verification algorithm $V(pk, m, s)$ returning 1 if $s = S(sk, m)$ or 0 otherwise.

The cryptographic system given by the triple (G, S, V) and their domains is called a signature scheme.

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **attack model**).

A security notion (or level) is entirely defined by coupling an adversarial goal with an adversarial model.

Examples: UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA.

GEMPLUS

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **attack model**).

A security notion (or level) is entirely defined by coupling an adversarial goal with an adversarial model.

Examples: UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA.

GEMPLUS

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **attack model**).

A security notion (or level) is entirely defined by coupling an adversarial goal with an adversarial model.

Examples: UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA.

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **attack model**).

A security notion (or level) is entirely defined by coupling an adversarial goal with an adversarial model.

Examples: UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA.

Security Goals

[Unbreakability] the attacker recovers the secret key sk from the public key pk (or an equivalent key if any). This goal is denoted **UB**. Implicitly appeared with public-key cryptography.

[Universal Unforgeability] the attacker, without necessarily having recovered sk, can produce a valid signature of any message in the message space. Noted **UUF**.

[Selective Unforgeability] the attacker can produce a valid signature of a message he committed to before knowing the public key. Noted **SUF**. Not often used in proofs (except in recent pairing-based signatures).

Security Goals

[Unbreakability] the attacker recovers the secret key *sk* from the public key *pk* (or an equivalent key if any). This goal is denoted **UB**. Implicitly appeared with public-key cryptography.

[Universal Unforgeability] the attacker, without necessarily having recovered *sk*, can produce a valid signature of any message in the message space. Noted **UUF**.

[Selective Unforgeability] the attacker can produce a valid signature of a message he committed to before knowing the public key. Noted **SUF**. Not often used in proofs (except in recent pairing-based signatures).

GEMPLUS

Security Goals

[Unbreakability] the attacker recovers the secret key *sk* from the public key *pk* (or an equivalent key if any). This goal is denoted **UB**. Implicitly appeared with public-key cryptography.

[Universal Unforgeability] the attacker, without necessarily having recovered *sk*, can produce a valid signature of any message in the message space. Noted **UUF**.

[Selective Unforgeability] the attacker can produce a valid signature of a message he committed to before knowing the public key. Noted **SUF**. Not often used in proofs (except in recent pairing-based signatures).

Security Goals

[Existential Unforgeability] the attacker creates a message and a valid signature of it (likely not of his choosing). Denoted **EUF**.

[Non-Malleability] the attacker is given (m, s) and is challenged to construct (m, s'). Denoted **NM**.

GEMPLUS

Security Goals

[Existential Unforgeability] the attacker creates a message and a valid signature of it (likely not of his choosing). Denoted **EUF**.

[Non-Malleability] the attacker is given (m, s) and is challenged to construct (m, s'). Denoted **NM**.

Adversarial Models

Several types of computational resources an adversary has access to are considered:

- **Key-Only Attacks** (KOA), unavoidable scenario.
- **Known Message Attacks** (KMA) where an adversary has access to signatures for a set of known messages.
- **Directed Chosen-Message Attacks** (DCMA) are a scenario in which the adversary chooses a set of messages $\{m_i\}_i$ and is given corresponding signatures $\{s_i\}_i$. The choice of $\{m_i\}_i$ is non-adaptive.

Adversarial Models

Several types of computational resources an adversary has access to are considered:

- **Key-Only Attacks** (KOA), unavoidable scenario.
- **Known Message Attacks** (KMA) where an adversary has access to signatures for a set of known messages.
- **Directed Chosen-Message Attacks** (DCMA) are a scenario in which the adversary chooses a set of messages $\{m_i\}_i$ and is given corresponding signatures $\{s_i\}_i$. The choice of $\{m_i\}_i$ is non-adaptive.

Adversarial Models

Several types of computational resources an adversary has access to are considered:

- **Key-Only Attacks** (KOA), unavoidable scenario.
- **Known Message Attacks** (KMA) where an adversary has access to signatures for a set of known messages.
- **Directed Chosen-Message Attacks** (DCMA) are a scenario in which the adversary chooses a set of messages $\{m_i\}_i$ and is given corresponding signatures $\{s_i\}_i$. The choice of $\{m_i\}_i$ is non-adaptive.

GEMPLUS

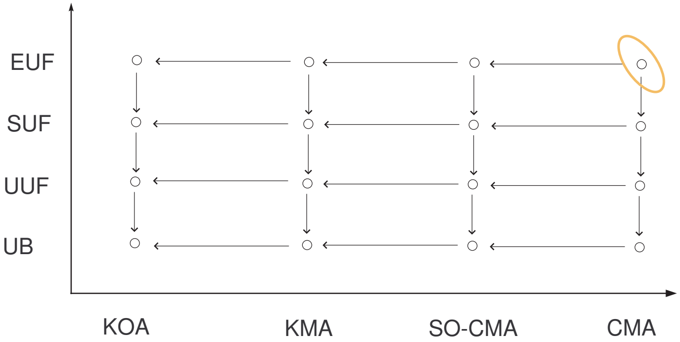
Adversarial Models (Cont'd)

- **Single Occurence Chosen-Message Attacks** (SOCMA) the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice but only once.
- **(Adaptive) Chosen-Message Attacks** (CMA) here too the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice (multiple requests of the same message are allowed).

Adversarial Models (Cont'd)

- **Single Occurence Chosen-Message Attacks** (SOCMA) the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice but only once.
- **(Adaptive) Chosen-Message Attacks** (CMA) here too the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice (multiple requests of the same message are allowed).

Relations Among Security Notions



Chosen-Message Security

Because EUF-CMA is the upper security level (Goldwasser, Micali, Rivest, 1988), it is desirable to prove security with respect to this notion.

Formally, an signature scheme is said to be (q, τ, ε)-secure if for any adversary $\mathcal{A}$ with running time upper-bounded by τ,

$$\mathrm{Succ}^{\mathrm{EUF-CMA}}(\mathcal{A}) = \Pr\left[\begin{array}{l} (sk, pk) \leftarrow G(1^k), \\ (m^*, s^*) \leftarrow \mathcal{A}^{S(sk,\cdot)}(pk), \\ V(pk, m^*, s^*) = 1 \end{array} \right] < \varepsilon \ ,$$

where the probability is taken over all random choices.

The notation $\mathcal{A}^{S(sk,\cdot)}$ means that the adversary has access to a signing oracle throughout the game, but at most q times.

The message m^* output by $\mathcal{A}$ must not have been requested to the signing oracle.

Chosen-Message Security

Because EUF-CMA is the upper security level (Goldwasser, Micali, Rivest, 1988), it is desirable to prove security with respect to this notion.

Formally, an signature scheme is said to be (q, τ, ε)-secure if for any adversary $\mathcal{A}$ with running time upper-bounded by τ,

$$\mathsf{Succ}^{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr\left[\begin{array}{l}(sk, pk) \leftarrow G(1^k),\\ (m^*, s^*) \leftarrow \mathcal{A}^{S(sk,\cdot)}(pk),\\ V(pk, m^*, s^*) = 1\end{array}\right] < \varepsilon\ ,$$

where the probability is taken over all random choices.

The notation $\mathcal{A}^{S(sk,\cdot)}$ means that the adversary has access to a signing oracle throughout the game, but at most q times.

The message m^* output by $\mathcal{A}$ must not have been requested to the signing oracle.

GEMPLUS

Chosen-Message Security

Because EUF-CMA is the upper security level (Goldwasser, Micali, Rivest, 1988), it is desirable to prove security with respect to this notion.

Formally, an signature scheme is said to be (q, τ, ε)-secure if for any adversary $\mathcal{A}$ with running time upper-bounded by τ,

$$\mathsf{Succ}^{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr\left[\begin{array}{l} (sk, pk) \leftarrow G(1^k), \\ (m^*, s^*) \leftarrow \mathcal{A}^{S(sk,\cdot)}(pk), \\ V(pk, m^*, s^*) = 1 \end{array}\right] < \varepsilon \ ,$$

where the probability is taken over all random choices.

The notation $\mathcal{A}^{S(sk,\cdot)}$ means that the adversary has access to a signing oracle throughout the game, but at most q times.

The message m^* output by $\mathcal{A}$ must not have been requested to the signing oracle.

Chosen-Message Security

Because EUF-CMA is the upper security level (Goldwasser, Micali, Rivest, 1988), it is desirable to prove security with respect to this notion.

Formally, an signature scheme is said to be (q, τ, ε)-secure if for any adversary $\mathcal{A}$ with running time upper-bounded by τ,

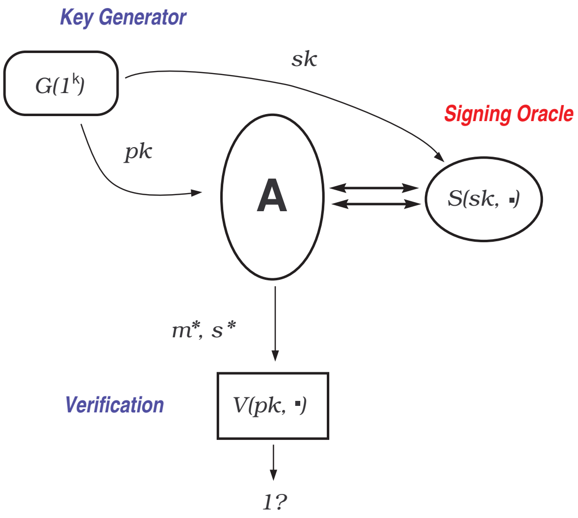
$$\mathsf{Succ}^{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr\left[\begin{array}{l}(sk, pk) \leftarrow G(1^k), \\ (m^*, s^*) \leftarrow \mathcal{A}^{S(sk,\cdot)}(pk), \\ V(pk, m^*, s^*) = 1\end{array}\right] < \varepsilon\ ,$$

where the probability is taken over all random choices.

The notation $\mathcal{A}^{S(sk,\cdot)}$ means that the adversary has access to a signing oracle throughout the game, but at most q times.

The message m^* output by $\mathcal{A}$ must not have been requested to the signing oracle.

EUF-CMA: Playing the Game



Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

We note $\mathcal{E}_{pk}(m) = \mathcal{E}_{pk}(m, \mathcal{U})$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

We note $\mathcal{E}_{pk}(m) = \mathcal{E}_{pk}(m, \mathcal{U})$.

GEMPLUS

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

We note $\mathcal{E}_{pk}(m) = \mathcal{E}_{pk}(m, \mathcal{U})$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

We note $\mathcal{E}_{pk}(m) = \mathcal{E}_{pk}(m, \mathcal{U})$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic decryption algorithm which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

We note $\mathcal{E}_{pk}(m) = \mathcal{E}_{pk}(m, \mathcal{U})$.

History of Security Goals

It shouldn't be feasible to:

- Compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitly appeared with public-key crypto.
- Invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- Recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- *Transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

History of Security Goals

It shouldn't be feasible to:

- Compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitly appeared with public-key crypto.
- Invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- Recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- *Transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

GEMPLUS

History of Security Goals

It shouldn't be feasible to:

- Compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitly appeared with public-key crypto.
- Invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- Recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- *Transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

GEMPLUS

History of Security Goals

It shouldn't be feasible to:

- Compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitly appeared with public-key crypto.
- Invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- Recover even a *single bit of information* about a plaintext given its encryption under any given key *pk* (indistinguishability of encryptions or IND). Goldwasser and Micali, 1984.
- *Transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

GEMPLUS

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario. Rackoff and Simon, 1991.

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

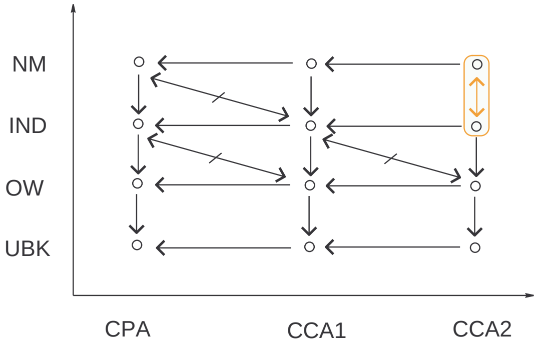
- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario. Rackoff and Simon, 1991.

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario. Rackoff and Simon, 1991.

Relations Among Security Notions



$\leftarrow$ indicates an implication: a scheme secure in notion A is also secure in notion B.

$\not\leftarrow$ indicates a separation: there exists a scheme secure in notion A but not in B.

Chosen-Ciphertext Security

Because IND-CCA2 $\equiv$ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

Formally, an asymmetric encryption scheme is said to be (τ, ε)-IND-CCA if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \xleftarrow{R} \{0,1\} \\ u \xleftarrow{R} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^\kappa), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \,,$$

where the probability is taken over the random choices of $\mathcal{A}$. The two plaintexts m_0 and m_1 chosen by the adversary have to be of identical length. Access to a decryption oracle is allowed throughout the game. We also have

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \left| \Pr[\mathcal{A} = 1 \mid b = 1] - \Pr[\mathcal{A} = 1 \mid b = 0] \right| \,.$$

GEMPLUS

Chosen-Ciphertext Security

Because IND-CCA2 $\equiv$ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

Formally, an asymmetric encryption scheme is said to be (τ, ε)-IND-CCA if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

$$\mathrm{Adv}^{\mathrm{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \stackrel{R}{\leftarrow} \{0,1\} \\ u \stackrel{R}{\leftarrow} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^{\kappa}), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \,,$$

where the probability is taken over the random choices of $\mathcal{A}$. The two plaintexts m_0 and m_1 chosen by the adversary have to be of identical length. Access to a decryption oracle is allowed throughout the game. We also have

$$\mathrm{Adv}^{\mathrm{ind}}(\mathcal{A}) = \left| \Pr[\mathcal{A} = 1 \mid b = 1] - \Pr[\mathcal{A} = 1 \mid b = 0] \right| .$$

Chosen-Ciphertext Security

Because IND-CCA2 $\equiv$ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

Formally, an asymmetric encryption scheme is said to be (τ, ε)-IND-CCA if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \overset{R}{\leftarrow} \{0,1\} \\ u \overset{R}{\leftarrow} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^\kappa), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \,,$$

where the probability is taken over the random choices of $\mathcal{A}$. The two plaintexts m_0 and m_1 chosen by the adversary have to be of identical length. Access to a decryption oracle is allowed throughout the game.

We also have

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \left| \Pr[\mathcal{A} = 1 \mid b = 1] - \Pr[\mathcal{A} = 1 \mid b = 0] \right| .$$

GEMPLUS

Chosen-Ciphertext Security

Because IND-CCA2 $\equiv$ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

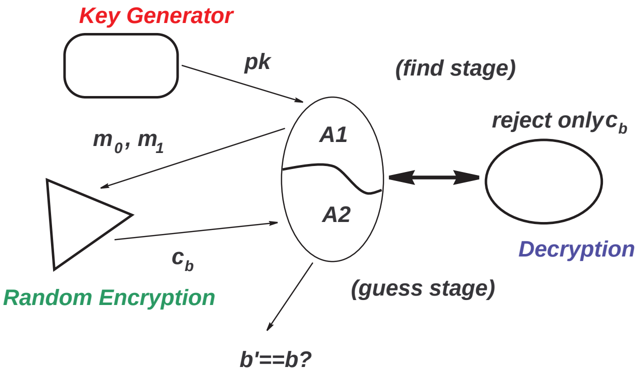
Formally, an asymmetric encryption scheme is said to be (τ, ε)-IND-CCA if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \xleftarrow{R} \{0,1\} \\ u \xleftarrow{R} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^\kappa), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \,,$$

where the probability is taken over the random choices of $\mathcal{A}$. The two plaintexts m_0 and m_1 chosen by the adversary have to be of identical length. Access to a decryption oracle is allowed throughout the game. We also have

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A} = 1 \mid b = 1\right] - \Pr\left[\mathcal{A} = 1 \mid b = 0\right] \right| .$$

IND-CCA: Playing the Game



How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers**. *But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

GEMPLUS

How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers**. *But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers**. *But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

GEMPLUS

How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers**. *But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers***. But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

GEMPLUS

How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers**. *But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

How Can We Build Cryptosystems?

These security notions are **targets** *for* **scheme designers***. But how does one design (secure) cryptosystems?*

Public-key design allows to construct systems by assembling and connecting smaller structures together. These may be *smaller* cryptosystems or atomic primitives:

- one-way functions, one-way trapdoor functions, one-way trapdoor permutations,
- hash functions, pseudo-random generators,
- secret-key permutations,
- message authentication codes,
- arithmetic or boolean operations, etc.

GEMPLUS

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, . . .

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, . . .

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

GEMPLUS

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

GEMPLUS

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Computational Assumptions

Cryptographic primitives are connected to plenty of (supposedly) intractable problems:

- RSA is one-way, Strong RSA is hard,
- discrete log is hard,
- computational/decisional Diffie-Hellman is hard,
- factoring is hard,
- shortest lattice vector is hard,
- computing residuosity classes is hard,
- deciding residuosity is hard, ...

Hard = Intractable = no PPT algorithm can solve the problem with non-negligible probability.

Schemes/Problems Reductions

Suppose we want to build some cryptosystem $\mathcal{S}$ and want a proof that (for instance)

$$\begin{aligned} RSA &\Leftarrow \text{EUF-CMA}(\mathcal{S}) && (1) \\ RSA &\Leftarrow \text{OW-CCA2}(\mathcal{E}) && (2) \end{aligned}$$

We have to show that breaking EUF-CMA($\mathcal{S}$) or OW-CCA2($\mathcal{E}$) allows to solve RSA, *i.e.* that an adversary breaking $\mathcal{S}$ can be used as a black box tool to answer RSA requests with non-negligible probability.

There is no such thing as a proof of security. There are only reductions

Probability Spaces: the reduction has to simulate the attacker's environment in a way that preserves (or does not alter too much) the distribution of all random variables which interact with it.

GEMPLUS

Schemes/Problems Reductions

Suppose we want to build some cryptosystem $\mathcal{S}$ and want a proof that (for instance)

$$\begin{aligned} RSA &\Leftarrow \text{EUF-CMA}(\mathcal{S}) && (1) \\ RSA &\Leftarrow \text{OW-CCA2}(\mathcal{E}) && (2) \end{aligned}$$

We have to show that breaking EUF-CMA($\mathcal{S}$) or OW-CCA2($\mathcal{E}$) allows to solve RSA, *i.e.* that an adversary breaking $\mathcal{S}$ can be used as a black box tool to answer RSA requests with non-negligible probability.

There is no such thing as a proof of security. There are only reductions

Probability Spaces: the reduction has to simulate the attacker's environment in a way that preserves (or does not alter too much) the distribution of all random variables which interact with it.

Schemes/Problems Reductions

Suppose we want to build some cryptosystem $\mathcal{S}$ and want a proof that (for instance)

$$RSA \Leftarrow \text{EUF-CMA}(\mathcal{S}) \quad (1)$$
$$RSA \Leftarrow \text{OW-CCA2}(\mathcal{E}) \quad (2)$$

We have to show that breaking EUF-CMA($\mathcal{S}$) or OW-CCA2($\mathcal{E}$) allows to solve RSA, *i.e.* that an adversary breaking $\mathcal{S}$ can be used as a black box tool to answer RSA requests with non-negligible probability.

There is no such thing as a proof of security. There are only reductions

Probability Spaces: the reduction has to simulate the attacker's environment in a way that preserves (or does not alter too much) the distribution of all random variables which interact with it.

GEMPLUS

Schemes/Problems Reductions

Suppose we want to build some cryptosystem $\mathcal{S}$ and want a proof that (for instance)

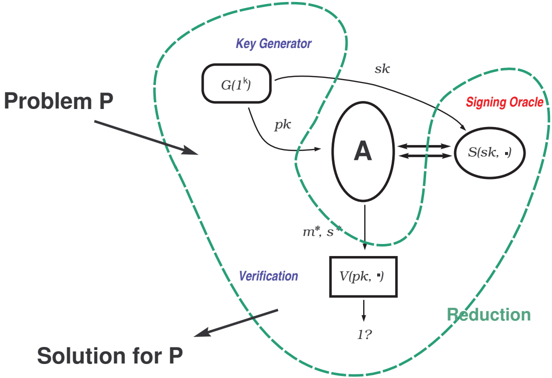
$$RSA \Leftarrow \text{EUF-CMA}(\mathcal{S}) \quad (1)$$
$$RSA \Leftarrow \text{OW-CCA2}(\mathcal{E}) \quad (2)$$

We have to show that breaking EUF-CMA($\mathcal{S}$) or OW-CCA2($\mathcal{E}$) allows to solve RSA, *i.e.* that an adversary breaking $\mathcal{S}$ can be used as a black box tool to answer RSA requests with non-negligible probability.

There is no such thing as a proof of security. There are only reductions

Probability Spaces: the reduction has to simulate the attacker's environment in a way that preserves (or does not alter too much) the distribution of all random variables which interact with it.

Simulating the Attacker's Environment



Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

For a practical impact, we need tight reductions to strong computational problems.

Some cryptosystems may feature asymptotic security but with an inefficient reduction ⇒ forces to use large keys ⇒ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

For a practical impact, we need tight reductions to strong computational problems.

Some cryptosystems may feature asymptotic security but with an inefficient reduction $\Rightarrow$ forces to use large keys $\Rightarrow$ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

For a practical impact, we need tight reductions to strong computational problems.

Some cryptosystems may feature asymptotic security but with an inefficient reduction ⇒ forces to use large keys ⇒ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

GEMPLUS

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

> *For a practical impact, we need tight reductions to strong computational problems.*

Some cryptosystems may feature asymptotic security but with an inefficient reduction ⇒ forces to use large keys ⇒ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

GEMPLUS

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

> *For a practical impact, we need tight reductions to strong computational problems.*

Some cryptosystems may feature asymptotic security but with an inefficient reduction $\Rightarrow$ forces to use large keys $\Rightarrow$ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

For a practical impact, we need tight reductions to strong computational problems.

Some cryptosystems may feature asymptotic security but with an inefficient reduction ⇛ forces to use large keys ⇛ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

> *For a practical impact, we need tight reductions to strong computational problems.*

Some cryptosystems may feature asymptotic security but with an inefficient reduction $\Rrightarrow$ forces to use large keys $\Rrightarrow$ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

Concrete Security

Provable security guarantees us that a scheme is asymptotically secure *i.e.* that all attacks asymptotically vanish thanks to polynomial reductions.

But what we need in real life is to provide explicit reductions.

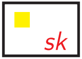
Exhibiting a reduction helps to decide how to tune the security parameter so that the scheme has a given concrete security.

> *For a practical impact, we need tight reductions to strong computational problems.*

Some cryptosystems may feature asymptotic security but with an inefficient reduction $\Rrightarrow$ forces to use large keys $\Rrightarrow$ heavier implementations: schemes may reveal useless. We need **tight** reductions so that we can guarantee security for **efficient** schemes.

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



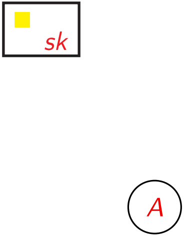
Smart Card

Decryption request

Signature request

Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*



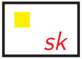
Smart Card

Decryption request

Signature request

Security Products with Top-Level Security

> *Security notions (goal + attack model) capture* *real-life* *attack scenarios. They really describe what we want.*



$E_{pk}(m)$

Smart Card

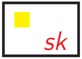
Decryption request

Signature request

A

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



$E_{pk}(m)$

Smart Card

Decryption request

Signature request

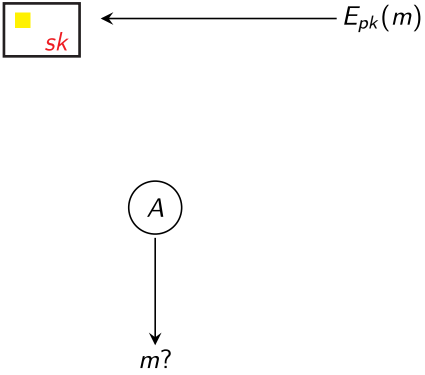
A

$\downarrow$

$m?$

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



Smart Card

Decryption request

Signature request

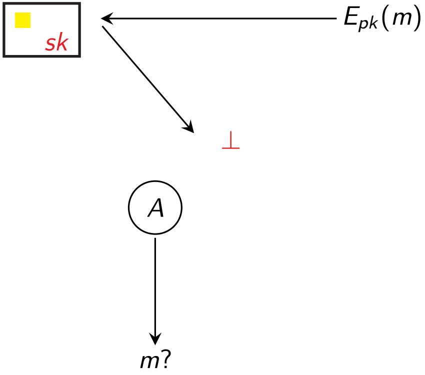
Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*

Smart Card

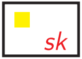
Decryption request

Signature request



Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



$E_{pk}(m)$

Smart Card

Decryption request

Signature request

$E_{pk}(m_1)$

A

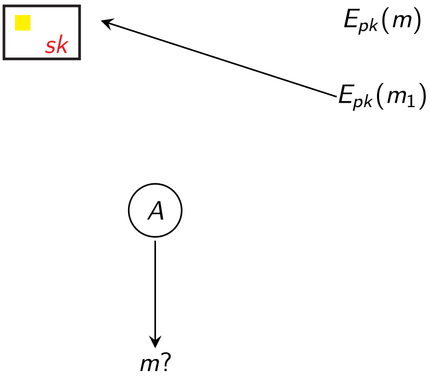
$\downarrow$

$m?$

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

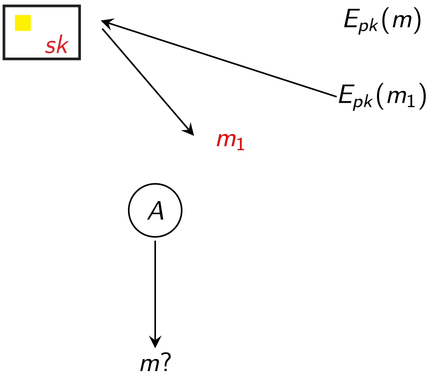
Smart Card
Decryption request
Signature request



Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

Smart Card
Decryption request
Signature request



Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*

sk

$E_{pk}(m)$

Smart Card

Decryption request

Signature request

$E_{pk}(m_1)$

$E_{pk}(m_2)$

A

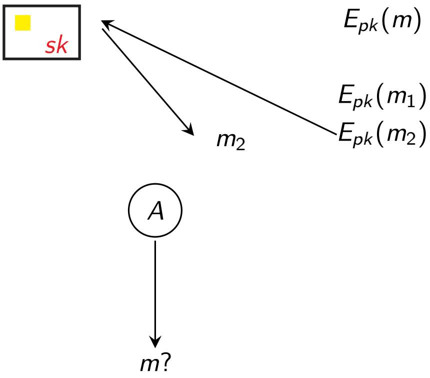
$\downarrow$

$m?$

Security Products with Top-Level Security

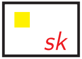
Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

Smart Card
Decryption request
Signature request



Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



$E_{pk}(m)$

Smart Card

Decryption request

Signature request

$E_{pk}(m_1)$

$E_{pk}(m_2)$

$\vdots$

$E_{pk}(m_n)$

A

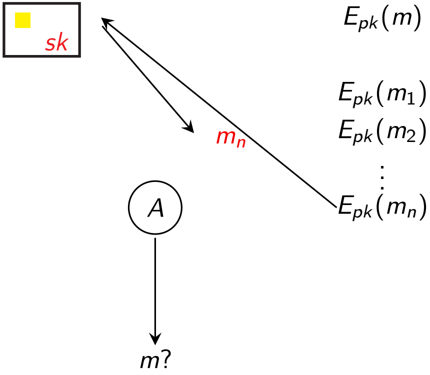
$\downarrow$

$m?$

Security Products with Top-Level Security

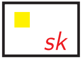
Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

Smart Card
Decryption request
Signature request



Security Products with Top-Level Security

> *Security notions (goal + attack model) capture* real-life *attack scenarios. They really describe what we want.*



$E_{pk}(m)$

Smart Card

Decryption request

Signature request

$E_{pk}(m_1)$
$E_{pk}(m_2)$
$\vdots$
$E_{pk}(m_n)$

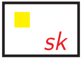
A

$\downarrow$

$m?$ not a clue!

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



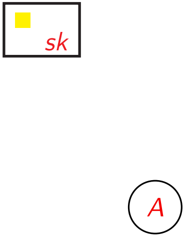
Smart Card

Decryption request

Signature request

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life *attack scenarios. They really describe what we want.*



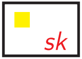
Smart Card

Decryption request

Signature request

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



$m =$ "You owe me \$1M"

Smart Card

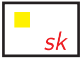
Decryption request

Signature request

A

Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*



$m =$ "You owe me \$1M"

Smart Card

Decryption request

Signature request

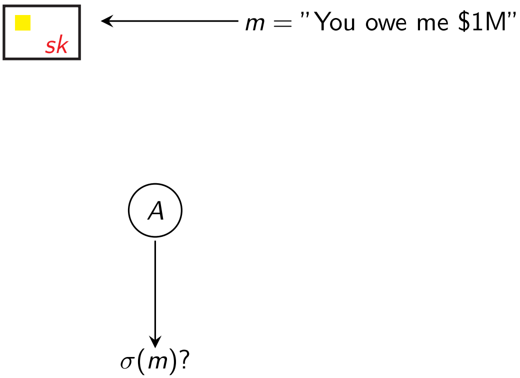
A

$\downarrow$

$\sigma(m)$?

Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*



Smart Card

Decryption request

Signature request

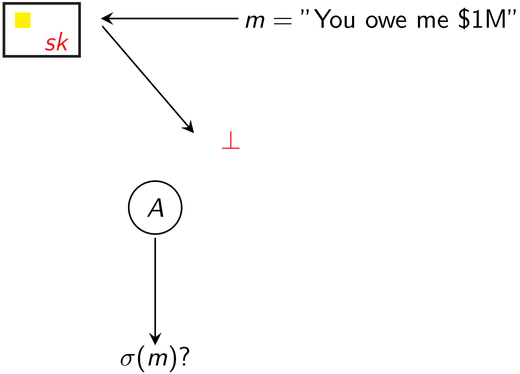
Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*

Smart Card

Decryption request

Signature request



Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*

sk

$m =$ "You owe me \$1M"

Smart Card

Decryption request

Signature request

m_1

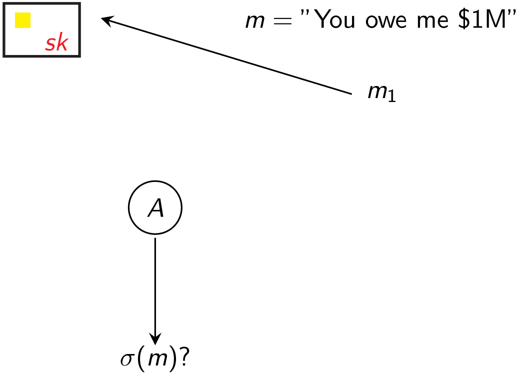
A

$\sigma(m)$?

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

Smart Card
Decryption request
Signature request



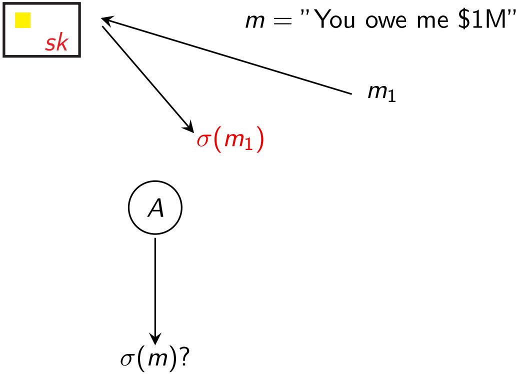
Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life *attack scenarios. They really describe what we want.*

Smart Card

Decryption request

Signature request



Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

sk

$m =$ "You owe me \$1M"

Smart Card

Decryption request

Signature request

m_1

m_2

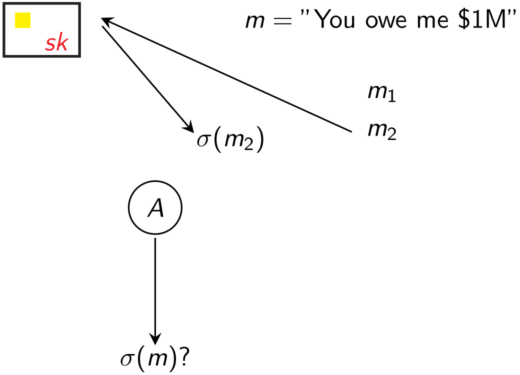
A

$\sigma(m)$?

Security Products with Top-Level Security

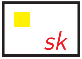
Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

Smart Card
Decryption request
Signature request



Security Products with Top-Level Security

> *Security notions (goal + attack model) capture* real-life *attack scenarios. They really describe what we want.*



$m =$ "You owe me \$1M"

Smart Card

Decryption request

Signature request

m_1

m_2

$\vdots$

m_n

A

$\downarrow$

$\sigma(m)$?

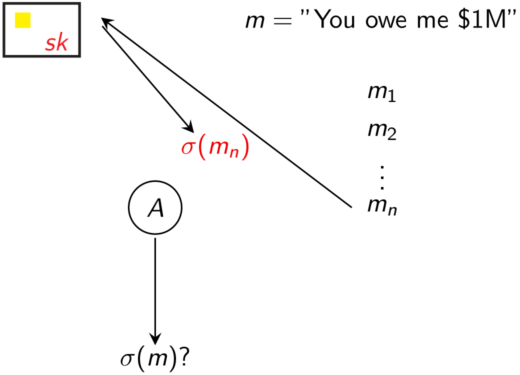
Security Products with Top-Level Security

> *Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.*

Smart Card

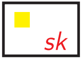
Decryption request

Signature request



Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.



$m =$ "You owe me \$1M"

Smart Card

Decryption request

Signature request

m_1

m_2

$\vdots$

m_n

A

$\sigma(m)$? not a clue!

Security Products with Top-Level Security

Security notions (goal + attack model) capture real-life attack scenarios. They really describe what we want.

sk

$m =$ "You owe me $1M"

Smart Card

Decryption request

Signature request

m_1

m_2

$\vdots$

m_n

A

But we need security proofs for that!

$\sigma(m)$?

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

NO: There exist schemes secure in the ROM which are insecure in the standard model!

It is a moral proof that spots design errors anyway...

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

NO: There exist schemes secure in the ROM which are insecure in the standard model!

YES: It is a moral proof that spots design errors anyway...

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

NO: There exist schemes secure in the ROM which are insecure in the standard model!

YES: It is a moral proof that spots design errors anyway...

GEMPLUS

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

NO: There exist schemes secure in the ROM which are insecure in the standard model!

YES: It is a moral proof that spots design errors anyway…

What Are Ideal Assumptions?

Providing reductions is rarely as easy as just seen. We often need to idealize our view of primitive objects in order to simplify the proof.

- ideal random hash functions $\rightrightarrows$ random oracle model,
- ideal symmetric encryption $\rightrightarrows$ ideal cipher model,
- ideal group $\rightrightarrows$ generic group model.

A reduction is easier between a given problem and a generic adversary!

Do people buy these proofs?

NO: There exist schemes secure in the ROM which are insecure in the standard model!

YES: It is a moral proof that spots design errors anyway...

Shoup's Modular Proofs

Security proofs are often **intricate** and details can be **implicit**. Important details of the proof may be overlooked (*e.g.* the OAEP saga).

Shoup introduced a proof design which facilitates public scrutiny.

The proof is given as a series of **rounds** or **games**.

The Difference (aka Shoup's) Lemma: Assume A, B, E are events and $\Pr[A \wedge \neg E] = \Pr[B \wedge \neg E]$. Then

$$|\Pr[A] - \Pr[B]| \leq \Pr[E] \ .$$

Shoup's Modular Proofs

Security proofs are often **intricate** and details can be **implicit**. Important details of the proof may be overlooked (*e.g.* the OAEP saga).

Shoup introduced a proof design which facilitates public scrutiny.

The proof is given as a series of **rounds** or **games**.

The Difference (aka Shoup's) Lemma: Assume A, B, E are events and $\Pr[A \wedge \neg E] = \Pr[B \wedge \neg E]$. Then

$$|\Pr[A] - \Pr[B]| \leq \Pr[E] \ .$$

GEMPLUS

Shoup's Modular Proofs

Security proofs are often **intricate** and details can be **implicit**. Important details of the proof may be overlooked (*e.g.* the OAEP saga).

Shoup introduced a proof design which facilitates public scrutiny.

The proof is given as a series of **rounds** or **games**.

The Difference (aka Shoup's) Lemma: Assume A, B, E are events and $\Pr[A \wedge \neg E] = \Pr[B \wedge \neg E]$. Then

$$|\Pr[A] - \Pr[B]| \leq \Pr[E] \ .$$

GEMPLUS

Shoup's Modular Proofs

Security proofs are often **intricate** and details can be **implicit**. Important details of the proof may be overlooked (*e.g.* the OAEP saga).

Shoup introduced a proof design which facilitates public scrutiny.

The proof is given as a series of **rounds** or **games**.

The Difference (aka Shoup's) Lemma: Assume A, B, E are events and $\Pr[A \wedge \neg E] = \Pr[B \wedge \neg E]$. Then

$$|\Pr[A] - \Pr[B]| \leq \Pr[E] \ .$$

Shoup's Modular Proofs

- the first game Game_0 is the one defined by the security model. **No reduction or simulations whatsoever**. The success probability $\Pr[S_0]$ of the adversary $\mathcal{A}$ is $\Pr[S_0] = \varepsilon_{\mathcal{A}}$.
- Game_{i+1} is described as being an **incrementally** modified version of Game_i. Then $\Pr[S_{i+1}]$ is expressed as a function of $\Pr[S_i]$ and scheme parameters.
- the last game Game_ℓ describes the complete reduction algorithm.

The last game provides $\varepsilon_{\mathcal{R}} = \Pr[S_\ell]$ as a function of $\Pr[S_0] = \varepsilon_{\mathcal{A}}$ and parameters. Execution time τ_ℓ is also expressed as a function of $\tau_0 = \tau_{\mathcal{A}}$.

Shoup's Modular Proofs

- the first game Game_0 is the one defined by the security model. **No reduction or simulations whatsoever**. The success probability $\Pr[S_0]$ of the adversary $\mathcal{A}$ is $\Pr[S_0] = \varepsilon_{\mathcal{A}}$.
- Game_{i+1} is described as being an **incrementally** modified version of Game_i. Then $\Pr[S_{i+1}]$ is expressed as a function of $\Pr[S_i]$ and scheme parameters.
- the last game Game_ℓ describes the complete reduction algorithm.

The last game provides $\varepsilon_{\mathcal{R}} = \Pr[S_\ell]$ as a function of $\Pr[S_0] = \varepsilon_{\mathcal{A}}$ and parameters. Execution time τ_ℓ is also expressed as a function of $\tau_0 = \tau_{\mathcal{A}}$.

Shoup's Modular Proofs

- the first game Game_0 is the one defined by the security model. **No reduction or simulations whatsoever**. The success probability $\Pr[S_0]$ of the adversary $\mathcal{A}$ is $\Pr[S_0] = \varepsilon_{\mathcal{A}}$.
- Game_{i+1} is described as being an **incrementally** modified version of Game_i. Then $\Pr[S_{i+1}]$ is expressed as a function of $\Pr[S_i]$ and scheme parameters.
- the last game Game_ℓ describes the complete reduction algorithm.

The last game provides $\varepsilon_{\mathcal{R}} = \Pr[S_\ell]$ as a function of $\Pr[S_0] = \varepsilon_{\mathcal{A}}$ and parameters. Execution time τ_ℓ is also expressed as a function of $\tau_0 = \tau_{\mathcal{A}}$.

Shoup's Modular Proofs

- the first game Game_0 is the one defined by the security model. **No reduction or simulations whatsoever**. The success probability $\Pr[S_0]$ of the adversary $\mathcal{A}$ is $\Pr[S_0] = \varepsilon_{\mathcal{A}}$.
- Game_{i+1} is described as being an **incrementally** modified version of Game_i. Then $\Pr[S_{i+1}]$ is expressed as a function of $\Pr[S_i]$ and scheme parameters.
- the last game Game_ℓ describes the complete reduction algorithm.

The last game provides $\varepsilon_{\mathcal{R}} = \Pr[S_\ell]$ as a function of $\Pr[S_0] = \varepsilon_{\mathcal{A}}$ and parameters. Execution time τ_ℓ is also expressed as a function of $\tau_0 = \tau_{\mathcal{A}}$.

Shoup's Modular Proofs

Adopting Shoup's methodology allows to

- check proofs more easily (longer proofs are possible),
- compare different proof strategies,
- concatenate proofs in a modular way by reusing pre-existing parts.

It makes it possible to build security reductions for cryptographic protocols that use provably secure ingredients.

Shoup's Modular Proofs

Adopting Shoup's methodology allows to

- check proofs more easily (longer proofs are possible),
- compare different proof strategies,
- concatenate proofs in a modular way by reusing pre-existing parts.

It makes it possible to build security reductions for cryptographic protocols that use provably secure ingredients.

Shoup's Modular Proofs

Adopting Shoup's methodology allows to

- check proofs more easily (longer proofs are possible),
- compare different proof strategies,
- concatenate proofs in a modular way by reusing pre-existing parts.

It makes it possible to build security reductions for cryptographic protocols that use provably secure ingredients.

GEMPLUS

Shoup's Modular Proofs

Adopting Shoup's methodology allows to

- check proofs more easily (longer proofs are possible),
- compare different proof strategies,
- concatenate proofs in a modular way by reusing pre-existing parts.

It makes it possible to build security reductions for cryptographic protocols that use provably secure ingredients.

The Ideal Cipher Model

Similar to the random oracle model, except that a **blockcipher** is replaced by a random permutation.

The random permutation E takes a pair (k, x) and returns $y = E(k; x)$. Of course $E^{-1}(k; y) = x$. Both E or E^{-1} may be queried.

A random permutation is easy to simulate: for any fresh pair (k, x), pick y at random such that $(k, x \leftrightarrow y) \notin \mathrm{Hist}[E]$ for any x, set $E(k; x) = y$ and return y. The history $\mathrm{Hist}[E]$ must be updated with the correspondence $(k, x \leftrightarrow y)$.

Open problem: is this equivalent to the random oracle model?

The Ideal Cipher Model

Similar to the random oracle model, except that a **blockcipher** is replaced by a random permutation.

The random permutation E takes a pair (k, x) and returns $y = E(k; x)$. Of course $E^{-1}(k; y) = x$. Both E or E^{-1} may be queried.

A random permutation is easy to simulate: for any fresh pair (k, x), pick y at random such that $(k, x \leftrightarrow y) \notin \text{Hist}[E]$ for any x, set $E(k; x) = y$ and return y. The history $\text{Hist}[E]$ must be updated with the correspondence $(k, x \leftrightarrow y)$.

Open problem: is this equivalent to the random oracle model?

GEMPLUS

The Ideal Cipher Model

Similar to the random oracle model, except that a **blockcipher** is replaced by a random permutation.

The random permutation E takes a pair (k, x) and returns $y = E(k; x)$. Of course $E^{-1}(k; y) = x$. Both E or E^{-1} may be queried.

A random permutation is easy to simulate: for any fresh pair (k, x), pick y at random such that $(k, x \leftrightarrow y) \notin \mathrm{Hist}\,[E]$ for any x, set $E(k; x) = y$ and return y. The history $\mathrm{Hist}\,[E]$ must be updated with the correspondence $(k, x \leftrightarrow y)$.

Open problem: is this equivalent to the random oracle model?

GEMPLUS

The Ideal Cipher Model

Similar to the random oracle model, except that a **blockcipher** is replaced by a random permutation.

The random permutation E takes a pair (k, x) and returns $y = E(k; x)$. Of course $E^{-1}(k; y) = x$. Both E or E^{-1} may be queried.

A random permutation is easy to simulate: for any fresh pair (k, x), pick y at random such that $(k, x \leftrightarrow y) \notin \mathrm{Hist}[E]$ for any x, set $E(k; x) = y$ and return y. The history $\mathrm{Hist}[E]$ must be updated with the correspondence $(k, x \leftrightarrow y)$.

Open problem: is this equivalent to the random oracle model?

The Generic Model

The generic model assumes that a given group G is ideal *i.e.* has no hidden structure behind the group structure.

No one can perform operations on group elements a, b other than group operations $c \leftarrow a \star b$, $c \leftarrow a^{-1}$ and test if $a \in G$.

All parties are provided with subroutines $\{\star, \cdot^{-1}, \text{test}\}$ that use their own representation of group elements as strings.

A proof standing in the generic model means that a successful adversary must exploit the structure of the group in a non classical fashion.

GEMPLUS

The Generic Model

The generic model assumes that a given group G is ideal *i.e.* has no hidden structure behind the group structure.

No one can perform operations on group elements a, b other than group operations $c \leftarrow a \star b$, $c \leftarrow a^{-1}$ and test if $a \in G$.

All parties are provided with subroutines $\{\star, \cdot^{-1}, \text{test}\}$ that use their own representation of group elements as strings.

A proof standing in the generic model means that a successful adversary must exploit the structure of the group in a non classical fashion.

The Generic Model

The generic model assumes that a given group G is ideal *i.e.* has no hidden structure behind the group structure.

No one can perform operations on group elements a, b *other than group operations* $c \leftarrow a \star b$, $c \leftarrow a^{-1}$ *and test if* $a \in G$*.*

All parties are provided with subroutines $\{\star, \cdot^{-1}, \text{test}\}$ that use their own representation of group elements as strings.

A proof standing in the generic model means that a successful adversary must exploit the structure of the group in a non classical fashion.

GEMPLUS

The Generic Model

The generic model assumes that a given group G is ideal *i.e.* has no hidden structure behind the group structure.

No one can perform operations on group elements a, b other than group operations $c \leftarrow a \star b$, $c \leftarrow a^{-1}$ and test if $a \in G$.

All parties are provided with subroutines $\{\star, \cdot^{-1}, \text{test}\}$ that use their own representation of group elements as strings.

A proof standing in the generic model means that a successful adversary must exploit the structure of the group in a non classical fashion.

GEMPLUS

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, ...): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, ...): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, ...): Various reductions in the ROM. Tight security reductions in the SM w/ weak problems.

Encryption Schemes

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM w/ weak problems.

Encryption Schemes

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, ...): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, ...): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, ...): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, ...), REACT, GEM I/II, ...): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash-Proof Systems (Cramer-Shoup, ...): Tight reduction in SM relative to a DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Ideas.

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, ...): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, ...): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, ...): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, ...), REACT, GEM I/II, ...): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash Proof Systems (Cramer-Shoup, ...): Tight reduction in SM relative to ≈DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Idem

GEMPLUS

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, . . .), REACT, GEM I/II, . . .): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash Proof Systems (Cramer-Shoup, . . .): Tight reduction in SM relative to ≈DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Idem.

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, . . .): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, . . .): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, . . .): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, . . .), REACT, GEM I/II, . . .): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash Proof Systems (Cramer-Shoup, . . .): Tight reduction in SM relative to ≈DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Idem.

GEMPLUS

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, ...): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, ...): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, ...): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, ...), REACT, GEM I/II, ...): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash Proof Systems (Cramer-Shoup, ...): Tight reduction in SM relative to $\approx$DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Idem.

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, ...): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, ...): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, ...): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, ...), REACT, GEM I/II, ...): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash Proof Systems (Cramer-Shoup, ...): Tight reduction in SM relative to ≈DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Idem.

Provable Security: Where Do We Stand From Now?

Signature Schemes

Hash-then-Sign (FDH, PSS/PSS-R, Esign, ...): Loose or tight reductions in the ROM. Nothing known in the Standard Model.

Classical Discrete-Log Based (Schnorr, ElGamal, DSA's, ...): No or loose reductions in the ROM. No security proofs in the SM.

Bilinear-Map-Based Schemes (Boneh-Boyen, ...): Various reductions in the ROM. Tight security reductions in the SM wrt weak problems.

Encryption Schemes

Ad-Hoc Conversions (OAEP(+, ...), REACT, GEM I/II, ...): Loose or tight reductions in the ROM. Nothing known in the SM.

Hash Proof Systems (Cramer-Shoup, ...): Tight reduction in SM relative to $\approx$DDH. Can we rely on stronger problems?

IBE-based Constructions (CHK, BCHK, BMW): Idem.

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account.

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account.

GEMPLUS

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account.

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account.

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account. Provably secure smart cards?

GEMPLUS

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account. Provably secure smart cards?

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account. Provably secure smart cards?

Provable Security: Trends

Convergence of Techniques. Proving equivalence of weakened proof models. Is it true that ROM $\equiv$ ICM?

Alleviate Proofs Models. Programmable vs. Non-programmable ROM/ICM/GGM. n-programmable oracles.

Getting Rid of These. ROM/ICM/GGM will become essentially pedagogical. Only the Standard Model will remain.

New Complexity Assumptions. New computational assumptions appear every year. Hope for a convergence towards simplified assumptions.

Impossibility Proofs. Proving that a security level cannot be reached due to weak design.

Optimality Proofs. Showing that a security reduction is optimal.

Physical Security. Taking side-channels and attacks by fault injection into account. Provably secure smart cards?

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs
- Automatic verification or generation of security proofs
- Extensions to the security of implementations of cryptosystems and protocols

Provable security is a rapidly evolving field. . .
but many challenging issues remain open

You are welcome to contribute the way you can

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field. . .
but many challenging issues remain open

You are welcome to contribute the way you can

GEMPLUS

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field. . .
but many challenging issues remain open

You are welcome to contribute the way you can

GEMPLUS

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field...
but many challenging issues remain open

You are welcome to contribute the way you can

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field...
but many challenging issues remain open

You are welcome to contribute the way you can

GEMPLUS

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field...
but many challenging issues remain open

You are welcome to contribute the way you can

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field...

but many challenging issues remain open

You are welcome to contribute the way you can

GEMPLUS

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field...
but many challenging issues remain open

You are welcome to contribute the way you can

The Holy Grail of Provable Security

- Cryptosystems with tight or perfect reductions wrt strong problems (factoring, dlog) in the Standard Model.
- Perfectly modular proofs so that composing cryptosystems/protocols simply means composing the proofs.
- Automatic verification or generation of security proofs.
- Extensions to the security of implementations of cryptosystems and protocols.

Provable security is a rapidly evolving field. . .
but many challenging issues remain open

You are welcome to contribute the way you can

GEMPLUS