Building and Attacking Lattice-Based Signatures

Part I: Construction With Fiat–Shamir

Mehdi Tibouchi

NTT Secure Platform Laboratories

Crypto-CO, Medellín, 2019–06–10

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

What's a digital signature?

- Like a real signature, but digital! (and more secure)
- Allows to authenticate a document
- Bob receives a document with Alice's signature: he is convinced that it comes from Alice, and can convince others
  - Alice has a private signing key she uses to compute the signature
  - She can publish the corresponding verification key that everyone can use to check the signature
- Contrast: message authentication codes (MACs): symmetric key, hence not transferable

©2019 NTT Secure Platform Laboratories

What's a digital signature?

- Like a real signature, but digital! (and more secure)
- Allows to authenticate a document
- Bob receives a document with Alice's signature: he is convinced that it comes from Alice, and can convince others
  - Alice has a private signing key she uses to compute the signature
  - She can publish the corresponding verification key that everyone can use to check the signature
- Contrast: message authentication codes (MACs): symmetric key, hence not transferable

©2019 NTT Secure Platform Laboratories

What's a digital signature?

- Like a real signature, but digital! (and more secure)
- Allows to authenticate a document
- Bob receives a document with Alice's signature: he is convinced that it comes from Alice, and can convince others
  - Alice has a private signing key she uses to compute the signature
  - She can publish the corresponding verification key that everyone can use to check the signature
- Contrast: message authentication codes (MACs): symmetric key, hence not transferable

©2019 NTT Secure Platform Laboratories

What's a digital signature?

- Like a real signature, but digital! (and more secure)
- Allows to authenticate a document
- Bob receives a document with Alice's signature: he is convinced that it comes from Alice, and can convince others
  - Alice has a private signing key she uses to compute the signature
  - She can publish the corresponding verification key that everyone can use to check the signature
- Contrast: message authentication codes (MACs): symmetric key, hence not transferable

©2019 NTT Secure Platform Laboratories

Uses of digital signatures (a few examples)

- Credit cards with chip-and-pin
  - The card signs each transaction
- Electronic tax returns (in France)
  - The taxpayer generates a digital signature on the form she submits
- Web server certificates (more generally: public key infrastructure)
  - How do I trust that the site called `amazon.com` is really what it seems, so I can send my payment info?
  - Because a "certificate authority" I trust says it is
  - Occasional problems with that model...
- Bitcoin and other cryptocurrencies
  - A Bitcoin address is the verification key of a digital signature;
  - A transaction from that address is accepted if it is signed with the corresponding signing key

©2019 NTT Secure Platform Laboratories

Uses of digital signatures (a few examples)

- Credit cards with chip-and-pin
  - The card signs each transaction
- Electronic tax returns (in France)
  - The taxpayer generates a digital signature on the form she submits
- Web server certificates (more generally: public key infrastructure)
  - How do I trust that the site called `amazon.com` is really what it seems, so I can send my payment info?
  - Because a "certificate authority" I trust says it is
  - Occasional problems with that model...
- Bitcoin and other cryptocurrencies
  - A Bitcoin address is the verification key of a digital signature;
  - A transaction from that address is accepted if it is signed with the corresponding signing key

©2019 NTT Secure Platform Laboratories

Uses of digital signatures (a few examples)

- Credit cards with chip-and-pin
  - The card signs each transaction
- Electronic tax returns (in France)
  - The taxpayer generates a digital signature on the form she submits
- Web server certificates (more generally: public key infrastructure)
  - How do I trust that the site called `amazon.com` is really what it seems, so I can send my payment info?
  - Because a "certificate authority" I trust says it is
  - Occasional problems with that model...
- Bitcoin and other cryptocurrencies
  - A Bitcoin address is the verification key of a digital signature;
  - A transaction from that address is accepted if it is signed with the corresponding signing key

©2019 NTT Secure Platform Laboratories

Uses of digital signatures (a few examples)

- Credit cards with chip-and-pin
  - The card signs each transaction
- Electronic tax returns (in France)
  - The taxpayer generates a digital signature on the form she submits
- Web server certificates (more generally: public key infrastructure)
  - How do I trust that the site called `amazon.com` is really what it seems, so I can send my payment info?
  - Because a "certificate authority" I trust says it is
  - Occasional problems with that model...
- Bitcoin and other cryptocurrencies
  - A Bitcoin address is the verification key of a digital signature;
  - A transaction from that address is accepted if it is signed with the corresponding signing key

©2019 NTT Secure Platform Laboratories

Formal definition

A digital signature scheme is a triple $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of probabilistic polynomial time (PPT) algorithms as follows:

- $\mathsf{KeyGen}(1^\lambda)$: samples a random key pair $(\mathsf{sk}, \mathsf{vk})$ consisting of a private signing key sk and a public verification key vk
- $\mathsf{Sign}(\mathsf{sk}, m)$: takes a message m in the message space (usually $\{0,1\}^*$) and the signing key, and outputs a signature σ
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: deterministic; outputs the bit 1 if σ is a valid signature on m, 0 otherwise

The scheme is correct if for all λ and all m:

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda);\\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m);\\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1\end{array}\right] = 1.$$

What about security?

©2019 NTT Secure Platform Laboratories

Formal definition

A digital signature scheme is a triple $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of probabilistic polynomial time (PPT) algorithms as follows:

- $\mathsf{KeyGen}(1^\lambda)$: samples a random key pair $(\mathsf{sk}, \mathsf{vk})$ consisting of a private signing key sk and a public verification key vk
- $\mathsf{Sign}(\mathsf{sk}, m)$: takes a message m in the message space (usually $\{0,1\}^*$) and the signing key, and outputs a signature σ
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: deterministic; outputs the bit 1 if σ is a valid signature on m, 0 otherwise

The scheme is correct if for all λ and all m:

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda);\\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m);\\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1\end{array}\right] = 1.$$

What about security?

©2019 NTT Secure Platform Laboratories

Formal definition

A digital signature scheme is a triple $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of probabilistic polynomial time (PPT) algorithms as follows:

- $\mathsf{KeyGen}(1^\lambda)$: samples a random key pair $(\mathsf{sk}, \mathsf{vk})$ consisting of a private signing key sk and a public verification key vk
- $\mathsf{Sign}(\mathsf{sk}, m)$: takes a message m in the message space (usually $\{0,1\}^*$) and the signing key, and outputs a signature σ
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$: deterministic; outputs the bit 1 if σ is a valid signature on m, 0 otherwise

The scheme is correct if for all λ and all m:

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^\lambda);\\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m);\\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1\end{array}\right] = 1.$$

What about security?

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Textbook RSA signatures

- One of the first proposals for digital signatures
- Signing key sk: two large primes p, q and a secret exponent d
- Verification key vk: (N, e) where $N = pq$ and $ed \equiv 1 \pmod{\phi(N)}$
- Signature on $m \in (\mathbb{Z}/N\mathbb{Z})^*$: $\sigma = m^d \bmod N$
- Verification: $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$
- Correctness: ok (Euler's totient theorem)
- Security?

©2019 NTT Secure Platform Laboratories

Are textbook RSA signatures secure?

- Recovering sk from vk is as hard as factoring N. Assuming factoring is hard, textbook RSA are therefore secure in some sense! ("unbreakability")
- However, many other problems:
  - Some messages are easy to sign (e.g. perfect e-th power $< N$)
  - After seen a valid signature-message pair (m, σ), can construct many more such pairs (m^k, σ^k)
  - Can sign arbitrary random messages: choose σ at random first, and let $m = \sigma^e \bmod N$
- So we should not consider the scheme secure!
- A good definition of security should capture all of those defects

©2019 NTT Secure Platform Laboratories

Are textbook RSA signatures secure?

- Recovering sk from vk is as hard as factoring N. Assuming factoring is hard, textbook RSA are therefore secure in some sense! ("unbreakability")
- However, many other problems:
  - Some messages are easy to sign (e.g. perfect e-th power $< N$)
  - After seen a valid signature-message pair (m, σ), can construct many more such pairs (m^k, σ^k)
  - Can sign arbitrary random messages: choose σ at random first, and let $m = \sigma^e \bmod N$
- So we should not consider the scheme secure!
- A good definition of security should capture all of those defects

©2019 NTT Secure Platform Laboratories

Are textbook RSA signatures secure?

- Recovering sk from vk is as hard as factoring N. Assuming factoring is hard, textbook RSA are therefore secure in some sense! ("unbreakability")
- However, many other problems:
  - Some messages are easy to sign (e.g. perfect e-th power $< N$)
  - After seen a valid signature-message pair (m, σ), can construct many more such pairs (m^k, σ^k)
  - Can sign arbitrary random messages: choose σ at random first, and let $m = \sigma^e \bmod N$
- So we should not consider the scheme secure!
- A good definition of security should capture all of those defects

©2019 NTT Secure Platform Laboratories

Are textbook RSA signatures secure?

- Recovering sk from vk is as hard as factoring N. Assuming factoring is hard, textbook RSA are therefore secure in some sense! ("unbreakability")
- However, many other problems:
  - Some messages are easy to sign (e.g. perfect e-th power $< N$)
  - After seen a valid signature-message pair (m, σ), can construct many more such pairs (m^k, σ^k)
  - Can sign arbitrary random messages: choose σ at random first, and let $m = \sigma^e \bmod N$
- So we should not consider the scheme secure!
- A good definition of security should capture all of those defects

©2019 NTT Secure Platform Laboratories

The right security definition for signatures: EUF-CMA

- We consider nowadays that the right security definition is the one proposed by Goldwasser–Micali–Rivest [GMR84]: existential unforgeability under chosen message attacks (EUF-CMA)
- An adversary (a PPT algorithm) tries to break the signature scheme
  - His goal (existential forgery) is to construct a valid signature σ^* on a message m^* of his choice, not seen before
  - His power (chosen message attack) is to ask the signer for valid signatures σ_i on arbitrarily many messages m_i that he chooses adaptively
- Clearly, textbook RSA signatures don't satisfy that definition!

©2019 NTT Secure Platform Laboratories

The right security definition for signatures: EUF-CMA

- We consider nowadays that the right security definition is the one proposed by Goldwasser–Micali–Rivest [GMR84]: existential unforgeability under chosen message attacks (EUF-CMA)
- An adversary (a PPT algorithm) tries to break the signature scheme
  - His goal (existential forgery) is to construct a valid signature σ^* on a message m^* of his choice, not seen before
  - His power (chosen message attack) is to ask the signer for valid signatures σ_i on arbitrarily many messages m_i that he chooses adaptively
- Clearly, textbook RSA signatures don't satisfy that definition!

©2019 NTT Secure Platform Laboratories

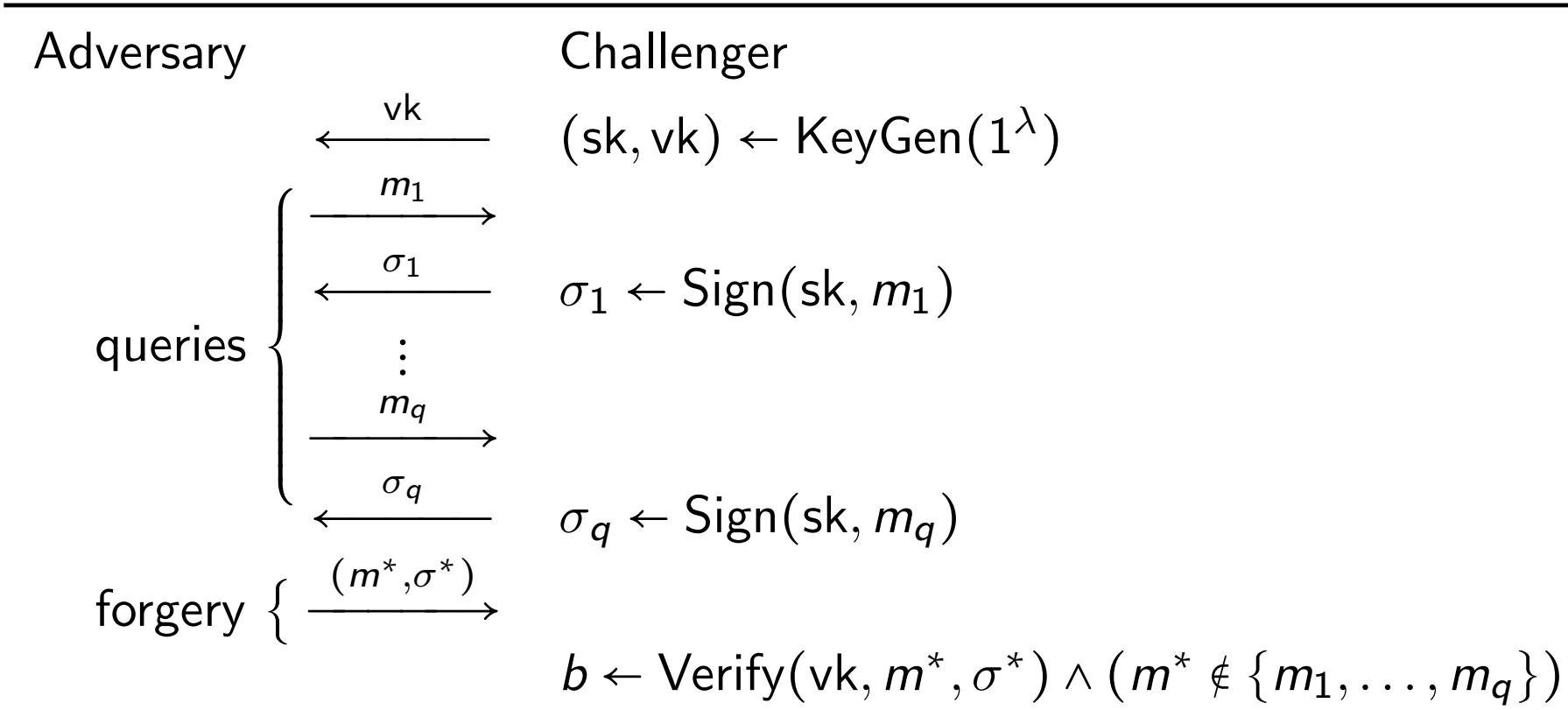
The right security definition for signatures: EUF-CMA

- We consider nowadays that the right security definition is the one proposed by Goldwasser–Micali–Rivest [GMR84]: existential unforgeability under chosen message attacks (EUF-CMA)
- An adversary (a PPT algorithm) tries to break the signature scheme
  - His goal (existential forgery) is to construct a valid signature σ^* on a message m^* of his choice, not seen before
  - His power (chosen message attack) is to ask the signer for valid signatures σ_i on arbitrarily many messages m_i that he chooses adaptively
- Clearly, textbook RSA signatures don't satisfy that definition!

©2019 NTT Secure Platform Laboratories

The EUF-CMA game

EUF-CMA security is modeled by the following game.



Adversary wins if $b = 1$. The scheme is EUF-CMA secure if the probability of any PPT adversary winning this game is negligible (i.e. $O(\lambda^{-c})$ for all $c > 0$).

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Identification protocols

- Remember that a signature scheme allows Alice to convince Bob (and everyone else) that she is the sender of a message that she puts her signature on
- Signatures are thus a form of non-interactive authentication: Bob does not need to talk to Alice to verify the signature (only needs her public verification key)
- Identification protocols are in some sense the interactive version of that
  - Alice still has a private key she keeps secret and a public key she shares with everyone
  - But now Bob interacts with Alice directly, and wants to make sure that he really talking to her and not someone else
  - Alice is the prover and Bob the verifier

©2019 NTT Secure Platform Laboratories

Identification protocols

- Remember that a signature scheme allows Alice to convince Bob (and everyone else) that she is the sender of a message that she puts her signature on
- Signatures are thus a form of non-interactive authentication: Bob does not need to talk to Alice to verify the signature (only needs her public verification key)
- Identification protocols are in some sense the interactive version of that
  - Alice still has a private key she keeps secret and a public key she shares with everyone
  - But now Bob interacts with Alice directly, and wants to make sure that he really talking to her and not someone else
  - Alice is the prover and Bob the verifier

©2019 NTT Secure Platform Laboratories

Identification protocols

- Remember that a signature scheme allows Alice to convince Bob (and everyone else) that she is the sender of a message that she puts her signature on
- Signatures are thus a form of non-interactive authentication: Bob does not need to talk to Alice to verify the signature (only needs her public verification key)
- Identification protocols are in some sense the interactive version of that
  - Alice still has a private key she keeps secret and a public key she shares with everyone
  - But now Bob interacts with Alice directly, and wants to make sure that he really talking to her and not someone else
  - Alice is the prover and Bob the verifier

©2019 NTT Secure Platform Laboratories

Security of an identification protocol

- The identification protocol is complete (or correct) if the verifier accepts the legitimate prover with overwhelming probability
- The protocol is secure against simple impersonation attacks if an adversary who sees only the public key can only convince the verifier with negligible probability
- The protocol is secure again full impersonation attacks (or eavesdropping attacks) if an adversary who sees the public key as well as arbitarily many interactions between the verifier and the legitimate prover can only convince the verifier with negligible probability
  - This is modeled by giving the adversary access to a transcript generation oracle
- Clearly, from an EUF-CMA secure signature scheme one can construct a 2-move ID protocol secure again full impersonation attacks

©2019 NTT Secure Platform Laboratories

Security of an identification protocol

- The identification protocol is complete (or correct) if the verifier accepts the legitimate prover with overwhelming probability
- The protocol is secure against simple impersonation attacks if an adversary who sees only the public key can only convince the verifier with negligible probability
- The protocol is secure again full impersonation attacks (or eavesdropping attacks) if an adversary who sees the public key as well as arbitarily many interactions between the verifier and the legitimate prover can only convince the verifier with negligible probability
  - This is modeled by giving the adversary access to a transcript generation oracle
- Clearly, from an EUF-CMA secure signature scheme one can construct a 2-move ID protocol secure again full impersonation attacks

©2019 NTT Secure Platform Laboratories

Security of an identification protocol

- The identification protocol is complete (or correct) if the verifier accepts the legitimate prover with overwhelming probability
- The protocol is secure against simple impersonation attacks if an adversary who sees only the public key can only convince the verifier with negligible probability
- The protocol is secure again full impersonation attacks (or eavesdropping attacks) if an adversary who sees the public key as well as arbitarily many interactions between the verifier and the legitimate prover can only convince the verifier with negligible probability
  - This is modeled by giving the adversary access to a transcript generation oracle
- Clearly, from an EUF-CMA secure signature scheme one can construct a 2-move ID protocol secure again full impersonation attacks

©2019 NTT Secure Platform Laboratories

Security of an identification protocol

- The identification protocol is complete (or correct) if the verifier accepts the legitimate prover with overwhelming probability
- The protocol is secure against simple impersonation attacks if an adversary who sees only the public key can only convince the verifier with negligible probability
- The protocol is secure again full impersonation attacks (or eavesdropping attacks) if an adversary who sees the public key as well as arbitarily many interactions between the verifier and the legitimate prover can only convince the verifier with negligible probability
  - This is modeled by giving the adversary access to a transcript generation oracle
- Clearly, from an EUF-CMA secure signature scheme one can construct a 2-move ID protocol secure again full impersonation attacks

©2019 NTT Secure Platform Laboratories

Identification from signatures

The prover's private key is his signing key sk in the signature scheme, and his public key is the verification key vk

Prover		Verifier
	$\xleftarrow{m}$	$m \leftarrow \{\text{message space}\}$
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$	$\xrightarrow{\sigma}$	
		accept iff $\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1$

Easy to see that if the signature scheme is EUF-CMA secure and the message space is large (e.g. of cardinality exponential in λ), this protocol is secure against full impersonation attacks

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Constructing identification protocols

- Since identification protocols are interactive, we expect them to be in some sense easier to construct that signatures
- In particular, there is a large family of protocols that have a particular shape; 3 moves as follows:
  1. the prover sends a commitment y to the verifier
  2. the verifier replies with a challenge c sampled uniformly from some challenge space
  3. the proved sends a response z to the challenge, consistent with its commitment
- These are called sigma protocols. The best-known example is the Schnorr identification scheme, based on the hardness of discrete logarithms in a cyclic group $G = \langle g \rangle$ of prime order p.

©2019 NTT Secure Platform Laboratories

Constructing identification protocols

- Since identification protocols are interactive, we expect them to be in some sense easier to construct that signatures
- In particular, there is a large family of protocols that have a particular shape; 3 moves as follows:
  1. the prover sends a commitment y to the verifier
  2. the verifier replies with a challenge c sampled uniformly from some challenge space
  3. the proved sends a response z to the challenge, consistent with its commitment
- These are called sigma protocols. The best-known example is the Schnorr identification scheme, based on the hardness of discrete logarithms in a cyclic group $G = \langle g \rangle$ of prime order p.

©2019 NTT Secure Platform Laboratories

Constructing identification protocols

- Since identification protocols are interactive, we expect them to be in some sense easier to construct that signatures
- In particular, there is a large family of protocols that have a particular shape; 3 moves as follows:
  1. the prover sends a commitment y to the verifier
  2. the verifier replies with a challenge c sampled uniformly from some challenge space
  3. the proved sends a response z to the challenge, consistent with its commitment
- These are called sigma protocols. The best-known example is the Schnorr identification scheme, based on the hardness of discrete logarithms in a cyclic group $G = \langle g \rangle$ of prime order p.

©2019 NTT Secure Platform Laboratories

Constructing identification protocols

- Since identification protocols are interactive, we expect them to be in some sense easier to construct that signatures
- In particular, there is a large family of protocols that have a particular shape; 3 moves as follows:
  1. the prover sends a commitment y to the verifier
  2. the verifier replies with a challenge c sampled uniformly from some challenge space
  3. the proved sends a response z to the challenge, consistent with its commitment
- These are called sigma protocols. The best-known example is the Schnorr identification scheme, based on the hardness of discrete logarithms in a cyclic group $G = \langle g \rangle$ of prime order p.

©2019 NTT Secure Platform Laboratories

Constructing identification protocols

- Since identification protocols are interactive, we expect them to be in some sense easier to construct that signatures
- In particular, there is a large family of protocols that have a particular shape; 3 moves as follows:
  1. the prover sends a commitment y to the verifier
  2. the verifier replies with a challenge c sampled uniformly from some challenge space
  3. the proved sends a response z to the challenge, consistent with its commitment
- These are called sigma protocols. The best-known example is the Schnorr identification scheme, based on the hardness of discrete logarithms in a cyclic group $G = \langle g \rangle$ of prime order p.

©2019 NTT Secure Platform Laboratories

Constructing identification protocols

- Since identification protocols are interactive, we expect them to be in some sense easier to construct that signatures
- In particular, there is a large family of protocols that have a particular shape; 3 moves as follows:
  1. the prover sends a commitment y to the verifier
  2. the verifier replies with a challenge c sampled uniformly from some challenge space
  3. the proved sends a response z to the challenge, consistent with its commitment
- These are called sigma protocols. The best-known example is the Schnorr identification scheme, based on the hardness of discrete logarithms in a cyclic group $G = \langle g \rangle$ of prime order p.

©2019 NTT Secure Platform Laboratories

Almost Schnorr

Prover's secret key: $x \leftarrow \{0, \ldots, p-1\}$. Public key: $h = g^x$.

Prover		Verifier
$r \leftarrow \{0, \ldots, p-1\}$; $y \leftarrow g^r$	$\xrightarrow{y}$	
	$\xleftarrow{c}$	$c \leftarrow \{0, \ldots, p-1\}$
$z \leftarrow r + cx$	$\xrightarrow{z}$	
		accepts iff $g^z = y \cdot h^c$ in G

This is almost, but not quite, the Schnorr protocol. This protocol is secure against simple impersonation attacks.

©2019 NTT Secure Platform Laboratories

Security: the forking lemma (I)

- Suppose that there exists a PPT adversary A that cheats the prover with non-negligible probability ε. We will prove that we can break the discrete logarithm problem in G with probability $\approx \varepsilon^2$
- We run this adversary until it is accepted by the prover, with probability ε. The transcript is then (y, c_1, z_1). Since this is an accepted transcript, we must have:

$$g^{z_1} = y \cdot h^{c_1}, \quad \text{hence} \quad y = g^{z_1 - x \cdot c_1}$$

©2019 NTT Secure Platform Laboratories

Security: the forking lemma (I)

- Suppose that there exists a PPT adversary A that cheats the prover with non-negligible probability ε. We will prove that we can break the discrete logarithm problem in G with probability $\approx \varepsilon^2$
- We run this adversary until it is accepted by the prover, with probability ε. The transcript is then (y, c_1, z_1). Since this is an accepted transcript, we must have:

$$g^{z_1} = y \cdot h^{c_1}, \quad \text{hence} \quad y = g^{z_1 - x \cdot c_1}$$

©2019 NTT Secure Platform Laboratories

Security: the forking lemma (II)

- Magic trick (Pointcheval–Stern): rewind the adversary just after its commitment y. On average, after this commitment, A still has a success probability of ε. We can thus obtain a second accepting transcript (y, c_2, z_2) with the same y! As above:

$$y = g^{z_2 - x \cdot c_2} = g^{z_1 - x \cdot c_1}$$

  therefore $z_1 - x \cdot c_1 \equiv z_2 - x \cdot c_2 \pmod{p}$.

- If $c_1 \neq c_2$, which happens except with probability $1/p$, we get

$$x = \frac{z_1 - z_2}{c_1 - c_2} \bmod p$$

  and we have solved the discrete log problem!

- Success probability $\geq \varepsilon \cdot (\varepsilon - 1/p)$

©2019 NTT Secure Platform Laboratories

Security: the forking lemma (II)

- Magic trick (Pointcheval–Stern): rewind the adversary just after its commitment y. On average, after this commitment, A still has a success probability of ε. We can thus obtain a second accepting transcript (y, c_2, z_2) with the same y! As above:

$$y = g^{z_2 - x \cdot c_2} = g^{z_1 - x \cdot c_1}$$

  therefore $z_1 - x \cdot c_1 \equiv z_2 - x \cdot c_2 \pmod{p}$.
- If $c_1 \neq c_2$, which happens except with probability $1/p$, we get

$$x = \frac{z_1 - z_2}{c_1 - c_2} \bmod p$$

  and we have solved the discrete log problem!
- Success probability $\geq \varepsilon \cdot (\varepsilon - 1/p)$

©2019 NTT Secure Platform Laboratories

Security: the forking lemma (II)

- Magic trick (Pointcheval–Stern): rewind the adversary just after its commitment y. On average, after this commitment, A still has a success probability of ε. We can thus obtain a second accepting transcript (y, c_2, z_2) with the same y! As above:

$$y = g^{z_2 - x \cdot c_2} = g^{z_1 - x \cdot c_1}$$

  therefore $z_1 - x \cdot c_1 \equiv z_2 - x \cdot c_2 \pmod{p}$.
- If $c_1 \neq c_2$, which happens except with probability $1/p$, we get

$$x = \frac{z_1 - z_2}{c_1 - c_2} \bmod p$$

  and we have solved the discrete log problem!
- Success probability $\geq \varepsilon \cdot (\varepsilon - 1/p)$

©2019 NTT Secure Platform Laboratories

What about full security?

- Problem to achieve full security?
- The response of the prover is $z = r + xc$, an integer in $[0, xp)$. The expectation of z is:

$$\mathbb{E}[z] = \frac{p-1}{2} + x\frac{p-1}{2} = \frac{(p-1)(x+1)}{2}.$$

- As a result, after seeing many executions of the protocol, an eavesdropper can learn (a good approximation of) x, and hence cheat the verifier!
- This does not happen in the real Schnorr protocol.

©2019 NTT Secure Platform Laboratories

What about full security?

- Problem to achieve full security?
- The response of the prover is $z = r + xc$, an integer in $[0, xp)$. The expectation of z is:

$$\mathbb{E}[z] = \frac{p-1}{2} + x\frac{p-1}{2} = \frac{(p-1)(x+1)}{2}.$$

- As a result, after seeing many executions of the protocol, an eavesdropper can learn (a good approximation of) x, and hence cheat the verifier!
- This does not happen in the real Schnorr protocol.

©2019 NTT Secure Platform Laboratories

What about full security?

- Problem to achieve full security?
- The response of the prover is $z = r + xc$, an integer in $[0, xp)$. The expectation of z is:

$$\mathbb{E}[z] = \frac{p-1}{2} + x\frac{p-1}{2} = \frac{(p-1)(x+1)}{2}.$$

- As a result, after seeing many executions of the protocol, an eavesdropper can learn (a good approximation of) x, and hence cheat the verifier!
- This does not happen in the real Schnorr protocol.

©2019 NTT Secure Platform Laboratories

What about full security?

- Problem to achieve full security?
- The response of the prover is $z = r + xc$, an integer in $[0, xp)$. The expectation of z is:

$$\mathbb{E}[z] = \frac{p-1}{2} + x\frac{p-1}{2} = \frac{(p-1)(x+1)}{2}.$$

- As a result, after seeing many executions of the protocol, an eavesdropper can learn (a good approximation of) x, and hence cheat the verifier!
- This does not happen in the real Schnorr protocol.

©2019 NTT Secure Platform Laboratories

Schnorr identification protocol

Prover's secret key: $x \leftarrow \{0, \ldots, p-1\}$. Public key: $h = g^x$.

Prover		Verifier
$r \leftarrow \{0, \ldots, p-1\}$; $y \leftarrow g^r$	$\xrightarrow{y}$	
	$\xleftarrow{c}$	$c \leftarrow \{0, \ldots, p-1\}$
$z \leftarrow r + cx \bmod p$	$\xrightarrow{z}$	
		accepts iff $g^z = y \cdot h^c$ in G

This protocol is secure against full impersonation attacks.

©2019 NTT Secure Platform Laboratories

Security: honest verifier zero-knowledge (I)

- Clearly, the reduction mod p prevents the previous attack. But how do we show that it prevents all attacks using valid transcripts?
- Idea: valid transcripts provide no information to an adversary, because they can be perfectly simulated with only the knowledge of the public key
- More precisely, correct transcripts are of the form (y, c, z) where $y = g^r$, r uniformly random in $\mathbb{Z}_p$; c uniformly random; and $z \in \mathbb{Z}_p$ uniquely defined by $g^z = y \cdot h^c$.

©2019 NTT Secure Platform Laboratories

Security: honest verifier zero-knowledge (I)

- Clearly, the reduction mod p prevents the previous attack. But how do we show that it prevents all attacks using valid transcripts?
- Idea: valid transcripts provide no information to an adversary, because they can be perfectly simulated with only the knowledge of the public key
- More precisely, correct transcripts are of the form (y, c, z) where $y = g^r$, r uniformly random in $\mathbb{Z}_p$; c uniformly random; and $z \in \mathbb{Z}_p$ uniquely defined by $g^z = y \cdot h^c$.

©2019 NTT Secure Platform Laboratories

Security: honest verifier zero-knowledge (I)

- Clearly, the reduction mod p prevents the previous attack. But how do we show that it prevents all attacks using valid transcripts?
- Idea: valid transcripts provide no information to an adversary, because they can be perfectly simulated with only the knowledge of the public key
- More precisely, correct transcripts are of the form (y, c, z) where $y = g^r$, r uniformly random in $\mathbb{Z}_p$; c uniformly random; and $z \in \mathbb{Z}_p$ uniquely defined by $g^z = y \cdot h^c$.

©2019 NTT Secure Platform Laboratories

Security: honest verifier zero-knowledge (II)

- We can simulate the previous distribution as follows:
  - c is uniformly random, so we pick it first;
  - then, for fixed c but arbitrary y, z is also uniformly random, so we pick it that way;
  - finally the correctness equation simply says $y = g^z \cdot h^{-c}$

  Hence, transcripts are distributed exactly like $(g^z/h^c, c, z)$ with uniformly random $c, z \in \mathbb{Z}_p$!
- Since the secret key is not necessary to generate valid transcripts, observing such transcripts gives no information to the adversary, hence full security

©2019 NTT Secure Platform Laboratories

Security: honest verifier zero-knowledge (II)

- We can simulate the previous distribution as follows:
  - c is uniformly random, so we pick it first;
  - then, for fixed c but arbitrary y, z is also uniformly random, so we pick it that way;
  - finally the correctness equation simply says $y = g^z \cdot h^{-c}$

  Hence, transcripts are distributed exactly like $(g^z/h^c, c, z)$ with uniformly random $c, z \in \mathbb{Z}_p$!
- Since the secret key is not necessary to generate valid transcripts, observing such transcripts gives no information to the adversary, hence full security

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

From identification to signatures

- We would like to turn these identification schemes using sigma protocols into signature schemes: make them non-interactive
- Note that the only interaction needed on the verifier's part is sending a uniformly random challenge between the two messages of the prover
- Fiat–Shamir heuristic: let's replace this interaction by a hash function modeled as a random oracle. Instead of asking the verifier for a random challenge, get it from an oracle query

©2019 NTT Secure Platform Laboratories

From identification to signatures

- We would like to turn these identification schemes using sigma protocols into signature schemes: make them non-interactive
- Note that the only interaction needed on the verifier's part is sending a uniformly random challenge between the two messages of the prover
- Fiat–Shamir heuristic: let's replace this interaction by a hash function modeled as a random oracle. Instead of asking the verifier for a random challenge, get it from an oracle query

©2019 NTT Secure Platform Laboratories

From identification to signatures

- We would like to turn these identification schemes using sigma protocols into signature schemes: make them non-interactive
- Note that the only interaction needed on the verifier's part is sending a uniformly random challenge between the two messages of the prover
- Fiat–Shamir heuristic: let's replace this interaction by a hash function modeled as a random oracle. Instead of asking the verifier for a random challenge, get it from an oracle query

©2019 NTT Secure Platform Laboratories

Schorr signature scheme

The Schnorr signature scheme is obtained from the Schnorr identification protocol via the Fiat–Shamir transform.

- KeyGen(1^λ): choose p and cyclic group $G = \langle g \rangle$ of order p according to the security parameter λ. H denote a random oracle $\{0,1\}^* \times G \to \mathbb{Z}_p$. The signing key sk is $x \leftarrow \mathbb{Z}_p$ uniformly random, and the verification key vk is $h = g^x$.
- Sign(sk, m): choose $r \in \mathbb{Z}_p$ uniformly at random. Let $y = g^r$, $c = H(m, y)$, and $z = r + cx \pmod p$. The signature is $\sigma = (y, z)$.
- Verify(vk, m, σ): compute $c = H(m, y)$, and accept iff $g^z = y \cdot h^c$.

©2019 NTT Secure Platform Laboratories

Schorr signature scheme

The Schnorr signature scheme is obtained from the Schnorr identification protocol via the Fiat–Shamir transform.

- KeyGen(1^λ): choose p and cyclic group $G = \langle g \rangle$ of order p according to the security parameter λ. H denote a random oracle $\{0,1\}^* \times G \to \mathbb{Z}_p$. The signing key sk is $x \leftarrow \mathbb{Z}_p$ uniformly random, and the verification key vk is $h = g^x$.
- Sign(sk, m): choose $r \in \mathbb{Z}_p$ uniformly at random. Let $y = g^r$, $c = H(m, y)$, and $z = r + cx \pmod p$. The signature is $\sigma = (y, z)$.
- Verify(vk, m, σ): compute $c = H(m, y)$, and accept iff $g^z = y \cdot h^c$.

©2019 NTT Secure Platform Laboratories

Schorr signature scheme

The Schnorr signature scheme is obtained from the Schnorr identification protocol via the Fiat–Shamir transform.

- KeyGen(1^λ): choose p and cyclic group $G = \langle g \rangle$ of order p according to the security parameter λ. H denote a random oracle $\{0,1\}^* \times G \to \mathbb{Z}_p$. The signing key sk is $x \leftarrow \mathbb{Z}_p$ uniformly random, and the verification key vk is $h = g^x$.
- Sign(sk, m): choose $r \in \mathbb{Z}_p$ uniformly at random. Let $y = g^r$, $c = H(m, y)$, and $z = r + cx \pmod p$. The signature is $\sigma = (y, z)$.
- Verify(vk, m, σ): compute $c = H(m, y)$, and accept iff $g^z = y \cdot h^c$.

©2019 NTT Secure Platform Laboratories

Security proof sketch

- We show that if there exists an adversary that breaks the signature scheme with non-negligible probability ε, one can construct a cheating prover that breaks the full security of the identification protocol with probability $\approx \varepsilon/q_h$ (q_h number of hashing queries of the adversary)
- Idea: guess the hashing query $H(m^*, y^*)$ corresponding to the adversary's forgery, use y^* as the cheating prover's commitment, and program the random oracle to set the hash value to the verifier's challenge c^*
- The guess is correct with probability $1/q_h$, hence the corresponding loss in success probability
- Signing queries are answered by using the transcript simulation technique
- There are additional negligible terms in the probability, to take into account unlikely bad events like hash query collisions

©2019 NTT Secure Platform Laboratories

Security proof sketch

- We show that if there exists an adversary that breaks the signature scheme with non-negligible probability ε, one can construct a cheating prover that breaks the full security of the identification protocol with probability $\approx \varepsilon/q_h$ (q_h number of hashing queries of the adversary)
- Idea: guess the hashing query $H(m^*, y^*)$ corresponding to the adversary's forgery, use y^* as the cheating prover's commitment, and program the random oracle to set the hash value to the verifier's challenge c^*
- The guess is correct with probability $1/q_h$, hence the corresponding loss in success probability
- Signing queries are answered by using the transcript simulation technique
- There are additional negligible terms in the probability, to take into account unlikely bad events like hash query collisions

©2019 NTT Secure Platform Laboratories

Security proof sketch

- We show that if there exists an adversary that breaks the signature scheme with non-negligible probability ε, one can construct a cheating prover that breaks the full security of the identification protocol with probability $\approx \varepsilon/q_h$ (q_h number of hashing queries of the adversary)
- Idea: guess the hashing query $H(m^*, y^*)$ corresponding to the adversary's forgery, use y^* as the cheating prover's commitment, and program the random oracle to set the hash value to the verifier's challenge c^*
- The guess is correct with probability $1/q_h$, hence the corresponding loss in success probability
- Signing queries are answered by using the transcript simulation technique
- There are additional negligible terms in the probability, to take into account unlikely bad events like hash query collisions

©2019 NTT Secure Platform Laboratories

Security proof sketch

- We show that if there exists an adversary that breaks the signature scheme with non-negligible probability ε, one can construct a cheating prover that breaks the full security of the identification protocol with probability $\approx \varepsilon/q_h$ (q_h number of hashing queries of the adversary)
- Idea: guess the hashing query $H(m^*, y^*)$ corresponding to the adversary's forgery, use y^* as the cheating prover's commitment, and program the random oracle to set the hash value to the verifier's challenge c^*
- The guess is correct with probability $1/q_h$, hence the corresponding loss in success probability
- Signing queries are answered by using the transcript simulation technique
- There are additional negligible terms in the probability, to take into account unlikely bad events like hash query collisions

©2019 NTT Secure Platform Laboratories

Security proof sketch

- We show that if there exists an adversary that breaks the signature scheme with non-negligible probability ε, one can construct a cheating prover that breaks the full security of the identification protocol with probability $\approx \varepsilon/q_h$ (q_h number of hashing queries of the adversary)
- Idea: guess the hashing query $H(m^*, y^*)$ corresponding to the adversary's forgery, use y^* as the cheating prover's commitment, and program the random oracle to set the hash value to the verifier's challenge c^*
- The guess is correct with probability $1/q_h$, hence the corresponding loss in success probability
- Signing queries are answered by using the transcript simulation technique
- There are additional negligible terms in the probability, to take into account unlikely bad events like hash query collisions

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

Lattice-based cryptography

- We will now propose an identification protocol using lattices, in the same style as Schnorr
- But we won't explain what a lattice is!
- Very roughly speaking, lattice-based cryptography is crypto based on linear algebra over $\mathbb{Z}$ or similar rings, together with a notion of "small" ring elements
- In our case, the ring will be $R = \mathbb{Z}_q[\mathbf{x}]/(\mathbf{x}^n + 1)$

©2019 NTT Secure Platform Laboratories

Lattice-based cryptography

- We will now propose an identification protocol using lattices, in the same style as Schnorr
- But we won't explain what a lattice is!
- Very roughly speaking, lattice-based cryptography is crypto based on linear algebra over $\mathbb{Z}$ or similar rings, together with a notion of "small" ring elements
- In our case, the ring will be $R = \mathbb{Z}_q[\mathbf{x}]/(\mathbf{x}^n + 1)$

©2019 NTT Secure Platform Laboratories

Lattice-based cryptography

- We will now propose an identification protocol using lattices, in the same style as Schnorr
- But we won't explain what a lattice is!
- Very roughly speaking, lattice-based cryptography is crypto based on linear algebra over $\mathbb{Z}$ or similar rings, together with a notion of "small" ring elements
- In our case, the ring will be $R = \mathbb{Z}_q[\mathbf{x}]/(\mathbf{x}^n + 1)$

©2019 NTT Secure Platform Laboratories

Lattice-based cryptography

- We will now propose an identification protocol using lattices, in the same style as Schnorr
- But we won't explain what a lattice is!
- Very roughly speaking, lattice-based cryptography is crypto based on linear algebra over $\mathbb{Z}$ or similar rings, together with a notion of "small" ring elements
- In our case, the ring will be $R = \mathbb{Z}_q[\mathbf{x}]/(\mathbf{x}^n + 1)$

©2019 NTT Secure Platform Laboratories

Hashing with Module-SIS

- The assumption we make is Module-SIS: for uniformly random elements $\mathbf{a}_1, \ldots, \mathbf{a}_m \in R$, it is hard to find $\mathbf{x}_1, \ldots, \mathbf{x}_m \in R$ with $\|\mathbf{x}\|_\infty \leq 2B$ such that

$$\mathbf{a}_1\mathbf{x}_1 + \cdots + \mathbf{a}_m\mathbf{x}_m = 0 \text{ in } R$$

- For simplicity, with denote vectors of elements of R with hats: $\hat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$.
- Clearly, we get another collision-resistant hash function family. Under the assumption above, the family $\{h_{\hat{\mathbf{a}}} \mid \hat{\mathbf{a}} \in R^m\}$ with

$$h_{\hat{\mathbf{a}}} \colon \{\hat{\mathbf{x}} \in R^m \mid \|\hat{\mathbf{x}}\|_\infty \leq B\} \to R, \quad \hat{\mathbf{x}} \mapsto \hat{\mathbf{a}} \cdot \hat{\mathbf{x}} = \sum \mathbf{a}_i \mathbf{x}_i$$

is collision-resistant

©2019 NTT Secure Platform Laboratories

Hashing with Module-SIS

- The assumption we make is Module-SIS: for uniformly random elements $\mathbf{a}_1, \ldots, \mathbf{a}_m \in R$, it is hard to find $\mathbf{x}_1, \ldots, \mathbf{x}_m \in R$ with $\|\mathbf{x}\|_\infty \leq 2B$ such that

$$\mathbf{a}_1\mathbf{x}_1 + \cdots + \mathbf{a}_m\mathbf{x}_m = 0 \text{ in } R$$

- For simplicity, with denote vectors of elements of R with hats: $\hat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$.
- Clearly, we get another collision-resistant hash function family. Under the assumption above, the family $\{h_{\hat{\mathbf{a}}} \mid \hat{\mathbf{a}} \in R^m\}$ with

$$h_{\hat{\mathbf{a}}} \colon \{\hat{\mathbf{x}} \in R^m \mid \|\hat{\mathbf{x}}\|_\infty \leq B\} \to R, \quad \hat{\mathbf{x}} \mapsto \hat{\mathbf{a}} \cdot \hat{\mathbf{x}} = \sum \mathbf{a}_i \mathbf{x}_i$$

is collision-resistant

©2019 NTT Secure Platform Laboratories

Hashing with Module-SIS

- The assumption we make is Module-SIS: for uniformly random elements $\mathbf{a}_1, \ldots, \mathbf{a}_m \in R$, it is hard to find $\mathbf{x}_1, \ldots, \mathbf{x}_m \in R$ with $\|\mathbf{x}\|_\infty \leq 2B$ such that

$$\mathbf{a}_1\mathbf{x}_1 + \cdots + \mathbf{a}_m\mathbf{x}_m = 0 \text{ in } R$$

- For simplicity, with denote vectors of elements of R with hats: $\hat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$.
- Clearly, we get another collision-resistant hash function family. Under the assumption above, the family $\{h_{\hat{\mathbf{a}}} \mid \hat{\mathbf{a}} \in R^m\}$ with

$$h_{\hat{\mathbf{a}}} \colon \{\hat{\mathbf{x}} \in R^m \mid \|\hat{\mathbf{x}}\|_\infty \leq B\} \to R, \quad \hat{\mathbf{x}} \mapsto \hat{\mathbf{a}} \cdot \hat{\mathbf{x}} = \sum \mathbf{a}_i\mathbf{x}_i$$

  is collision-resistant

©2019 NTT Secure Platform Laboratories

Identification protocol from Module-SIS (I)

Prover's secret key: $\hat{\mathbf{s}} \in R^m$ with $\|\hat{\mathbf{s}}\|_\infty \leq \sigma$. Public key: h in the hash family, $\mathbf{S} = h(\hat{\mathbf{s}})$.

Prover		Verifier
$\hat{\mathbf{y}} \leftarrow R^m$ with $\|\hat{\mathbf{y}}\|_\infty \leq \alpha$		
$\mathbf{Y} \leftarrow h(\hat{\mathbf{y}})$	$\xrightarrow{\mathbf{Y}}$	
	$\xleftarrow{\mathbf{c}}$	$\mathbf{c} \leftarrow R$ with $\|\mathbf{c}\|_1 = \kappa$
$\hat{\mathbf{z}} \leftarrow \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$	$\xrightarrow{\hat{\mathbf{z}}}$	
		accepts iff $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Simple security with the forking lemma

- The previous scheme is secure against simple impersonation attacks
- Idea: use the forking lemma again. If a cheating prover wins with probability ε, then with probability $\approx \varepsilon^2$, we obtain two accepting transcripts $(\mathbf{Y}, \mathbf{c}_1, \hat{\mathbf{z}}_1)$, $(\mathbf{Y}, \mathbf{c}_2, \hat{\mathbf{z}}_2)$
- Validity implies:

$$\mathbf{Y} = h(\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1) = h(\hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2),$$

  and we can assume that the simulator knows the secret key $\hat{\mathbf{s}}$. Thus, we obtain a collision on h, and break collision resistance!
- Subtlety to take into account: what if $\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1 = \hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2$? Can be avoided with a witness indistinguishability argument: with high probability, there is another secret key $\hat{\mathbf{s}}'$ giving the same public key $\mathbf{S}$

©2019 NTT Secure Platform Laboratories

Simple security with the forking lemma

- The previous scheme is secure against simple impersonation attacks
- Idea: use the forking lemma again. If a cheating prover wins with probability ε, then with probability $\approx \varepsilon^2$, we obtain two accepting transcripts $(\mathbf{Y}, \mathbf{c}_1, \hat{\mathbf{z}}_1)$, $(\mathbf{Y}, \mathbf{c}_2, \hat{\mathbf{z}}_2)$
- Validity implies:

$$\mathbf{Y} = h(\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1) = h(\hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2),$$

  and we can assume that the simulator knows the secret key $\hat{\mathbf{s}}$. Thus, we obtain a collision on h, and break collision resistance!
- Subtlety to take into account: what if $\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1 = \hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2$? Can be avoided with a witness indistinguishability argument: with high probability, there is another secret key $\hat{\mathbf{s}}'$ giving the same public key $\mathbf{S}$

©2019 NTT Secure Platform Laboratories

Simple security with the forking lemma

- The previous scheme is secure against simple impersonation attacks
- Idea: use the forking lemma again. If a cheating prover wins with probability ε, then with probability $\approx \varepsilon^2$, we obtain two accepting transcripts $(\mathbf{Y}, \mathbf{c}_1, \hat{\mathbf{z}}_1)$, $(\mathbf{Y}, \mathbf{c}_2, \hat{\mathbf{z}}_2)$
- Validity implies:

$$\mathbf{Y} = h(\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1) = h(\hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2),$$

  and we can assume that the simulator knows the secret key $\hat{\mathbf{s}}$. Thus, we obtain a collision on h, and break collision resistance!
- Subtlety to take into account: what if $\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1 = \hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2$? Can be avoided with a witness indistinguishability argument: with high probability, there is another secret key $\hat{\mathbf{s}}'$ giving the same public key $\mathbf{S}$

©2019 NTT Secure Platform Laboratories

Simple security with the forking lemma

- The previous scheme is secure against simple impersonation attacks
- Idea: use the forking lemma again. If a cheating prover wins with probability ε, then with probability $\approx \varepsilon^2$, we obtain two accepting transcripts $(\mathbf{Y}, \mathbf{c}_1, \hat{\mathbf{z}}_1)$, $(\mathbf{Y}, \mathbf{c}_2, \hat{\mathbf{z}}_2)$
- Validity implies:

$$\mathbf{Y} = h(\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1) = h(\hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2),$$

  and we can assume that the simulator knows the secret key $\hat{\mathbf{s}}$. Thus, we obtain a collision on h, and break collision resistance!
- Subtlety to take into account: what if $\hat{\mathbf{z}}_1 - \hat{\mathbf{s}}\mathbf{c}_1 = \hat{\mathbf{z}}_2 - \hat{\mathbf{s}}\mathbf{c}_2$? Can be avoided with a witness indistinguishability argument: with high probability, there is another secret key $\hat{\mathbf{s}}'$ giving the same public key $\mathbf{S}$

©2019 NTT Secure Platform Laboratories

Outline

Digital signatures
- Definition, applications
- Security of digital signatures

The Fiat–Shamir paradigm
- Identification protocols
- The Schnorr identification protocol
- Fiat–Shamir

Lattice-based Fiat–Shamir signatures
- Identification protocol from Module-SIS
- Fiat–Shamir with aborts

©2019 NTT Secure Platform Laboratories

What about full security?

- Is the protocol fully secure?
- No: same problem as in our Almost Schnorr protocol. The prover's response $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$ has a distribution depending on his secret key
- From many executions of the protocol, can recover the a good approximation of the key!
- Solution: aborting in some cases to make the distribution independent of $\hat{\mathbf{s}}$

©2019 NTT Secure Platform Laboratories

What about full security?

- Is the protocol fully secure?
- No: same problem as in our Almost Schnorr protocol. The prover's response $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$ has a distribution depending on his secret key
- From many executions of the protocol, can recover the a good approximation of the key!
- Solution: aborting in some cases to make the distribution independent of $\hat{\mathbf{s}}$

©2019 NTT Secure Platform Laboratories

What about full security?

- Is the protocol fully secure?
- No: same problem as in our Almost Schnorr protocol. The prover's response $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$ has a distribution depending on his secret key
- From many executions of the protocol, can recover the a good approximation of the key!
- Solution: aborting in some cases to make the distribution independent of $\hat{\mathbf{s}}$

©2019 NTT Secure Platform Laboratories

What about full security?

- Is the protocol fully secure?
- No: same problem as in our Almost Schnorr protocol. The prover's response $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$ has a distribution depending on his secret key
- From many executions of the protocol, can recover the a good approximation of the key!
- Solution: aborting in some cases to make the distribution independent of $\hat{\mathbf{s}}$

©2019 NTT Secure Platform Laboratories

Identification protocol from Module-SIS (II)

Prover's secret key: $\hat{\mathbf{s}} \in R^m$ with $\|\hat{\mathbf{s}}\|_\infty \leq \sigma$. Public key: h in the hash family, $\mathbf{S} = h(\mathbf{s})$.

Prover		Verifier
$\hat{\mathbf{y}} \leftarrow R^m$ with $\|\hat{\mathbf{y}}\|_\infty \leq \alpha$		
$\mathbf{Y} \leftarrow h(\hat{\mathbf{y}})$	$\xrightarrow{\mathbf{Y}}$	
	$\xleftarrow{\mathbf{c}}$	$\mathbf{c} \leftarrow R$ with $\|\mathbf{c}\|_1 = \kappa$
$\hat{\mathbf{z}} \leftarrow \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$		
if $\|\hat{\mathbf{z}}\|_\infty > \alpha - \sigma\kappa$, $\hat{\mathbf{z}} \leftarrow \perp$	$\xrightarrow{\hat{\mathbf{z}}}$	
		accepts iff $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Argument for full security

- In the cases when $\hat{\mathbf{z}} \neq \perp$ (the prover does not abort), then the distribution of $\hat{\mathbf{z}}$ is uniform in $\{\hat{\mathbf{u}} \in R^m \mid \|\mathbf{u}\|_\infty \leq \alpha - \sigma\kappa\}$
- Therefore, an adversary cannot learn anything from those transcripts (can be simulated without the secret key)
- Rejection probability also does not depend on $\hat{\mathbf{s}}$
- Issue with aborted transcripts: are they really indepedent of $\hat{\mathbf{s}}$? Not clear, but in this case, we erase the history

©2019 NTT Secure Platform Laboratories

Argument for full security

- In the cases when $\hat{\mathbf{z}} \neq \perp$ (the prover does not abort), then the distribution of $\hat{\mathbf{z}}$ is uniform in $\{\hat{\mathbf{u}} \in R^m \mid \|\mathbf{u}\|_\infty \leq \alpha - \sigma\kappa\}$
- Therefore, an adversary cannot learn anything from those transcripts (can be simulated without the secret key)
- Rejection probability also does not depend on $\hat{\mathbf{s}}$
- Issue with aborted transcripts: are they really indepedent of $\hat{\mathbf{s}}$? Not clear, but in this case, we erase the history

©2019 NTT Secure Platform Laboratories

Argument for full security

- In the cases when $\hat{\mathbf{z}} \neq \perp$ (the prover does not abort), then the distribution of $\hat{\mathbf{z}}$ is uniform in $\{\hat{\mathbf{u}} \in R^m \mid \|\mathbf{u}\|_\infty \leq \alpha - \sigma\kappa\}$
- Therefore, an adversary cannot learn anything from those transcripts (can be simulated without the secret key)
- Rejection probability also does not depend on $\hat{\mathbf{s}}$
- Issue with aborted transcripts: are they really indepedent of $\hat{\mathbf{s}}$? Not clear, but in this case, we erase the history

©2019 NTT Secure Platform Laboratories

Argument for full security

- In the cases when $\hat{\mathbf{z}} \neq \perp$ (the prover does not abort), then the distribution of $\hat{\mathbf{z}}$ is uniform in $\{\hat{\mathbf{u}} \in R^m \mid \|\mathbf{u}\|_\infty \leq \alpha - \sigma\kappa\}$
- Therefore, an adversary cannot learn anything from those transcripts (can be simulated without the secret key)
- Rejection probability also does not depend on $\hat{\mathbf{s}}$
- Issue with aborted transcripts: are they really indepedent of $\hat{\mathbf{s}}$? Not clear, but in this case, we erase the history

©2019 NTT Secure Platform Laboratories

Applying Fiat–Shamir

The signature scheme obtained from the previous identification protocol via the Fiat–Shamir transform is as follows.

- KeyGen(1^λ): set R and the parameters $m, \alpha, \kappa, \sigma$ according to λ. H is a random oracle $\{0,1\}^* \times R \to D_c$. Sample $\hat{\mathbf{s}}$ as the signing key, h and $\mathbf{S} = h(\hat{\mathbf{s}})$ as the verification key.
- Sign$(\hat{\mathbf{s}}, h, \mu)$:
  - sample $\hat{\mathbf{y}}$ and let $\mathbf{Y} = h(\hat{\mathbf{y}})$
  - let $\mathbf{c} = H(\mu, \mathbf{Y})$ and $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$
  - if $\|\hat{\mathbf{z}}\| > \alpha - \sigma\kappa$ restart, else output sig $= (\mathbf{Y}, \hat{\mathbf{z}})$
- Verify$\big(h, \mathbf{S}, \mu, \text{sig} = (\mathbf{Y}, \hat{\mathbf{z}})\big)$:
  - compute $\mathbf{c} = H(\mu, \mathbf{Y})$
  - accept iff $\|\hat{\mathbf{z}}\| \leq \alpha - \sigma\kappa$ and $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Applying Fiat–Shamir

The signature scheme obtained from the previous identification protocol via the Fiat–Shamir transform is as follows.

- KeyGen(1^λ): set R and the parameters $m, \alpha, \kappa, \sigma$ according to λ. H is a random oracle $\{0,1\}^* \times R \to D_c$. Sample $\hat{\mathbf{s}}$ as the signing key, h and $\mathbf{S} = h(\hat{\mathbf{s}})$ as the verification key.
- Sign$(\hat{\mathbf{s}}, h, \mu)$:
  - sample $\hat{\mathbf{y}}$ and let $\mathbf{Y} = h(\hat{\mathbf{y}})$
  - let $\mathbf{c} = H(\mu, \mathbf{Y})$ and $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$
  - if $\|\hat{\mathbf{z}}\| > \alpha - \sigma\kappa$ restart, else output sig $= (\mathbf{Y}, \hat{\mathbf{z}})$
- Verify$\big(h, \mathbf{S}, \mu, \text{sig} = (\mathbf{Y}, \hat{\mathbf{z}})\big)$:
  - compute $\mathbf{c} = H(\mu, \mathbf{Y})$
  - accept iff $\|\hat{\mathbf{z}}\| \leq \alpha - \sigma\kappa$ and $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories

Applying Fiat–Shamir

The signature scheme obtained from the previous identification protocol via the Fiat–Shamir transform is as follows.

- KeyGen(1^λ): set R and the parameters $m, \alpha, \kappa, \sigma$ according to λ. H is a random oracle $\{0,1\}^* \times R \to D_c$. Sample $\hat{\mathbf{s}}$ as the signing key, h and $\mathbf{S} = h(\hat{\mathbf{s}})$ as the verification key.
- Sign$(\hat{\mathbf{s}}, h, \mu)$:
  - sample $\hat{\mathbf{y}}$ and let $\mathbf{Y} = h(\hat{\mathbf{y}})$
  - let $\mathbf{c} = H(\mu, \mathbf{Y})$ and $\hat{\mathbf{z}} = \hat{\mathbf{s}}\mathbf{c} + \hat{\mathbf{y}}$
  - if $\|\hat{\mathbf{z}}\| > \alpha - \sigma\kappa$ restart, else output sig $= (\mathbf{Y}, \hat{\mathbf{z}})$
- Verify$\big(h, \mathbf{S}, \mu, \text{sig} = (\mathbf{Y}, \hat{\mathbf{z}})\big)$:
  - compute $\mathbf{c} = H(\mu, \mathbf{Y})$
  - accept iff $\|\hat{\mathbf{z}}\| \leq \alpha - \sigma\kappa$ and $h(\hat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

©2019 NTT Secure Platform Laboratories