Hash-Based Signatures.
Authenticated Key-Exchange using Digital Signatures.

Applied Cryptography – Spring 2022

Simona Samardjiska

May 11, 2022

Institute for Computing and Information Sciences
Radboud University

Summary

Last time:

- Introduction to Post-Quantum Cryptography

Today:

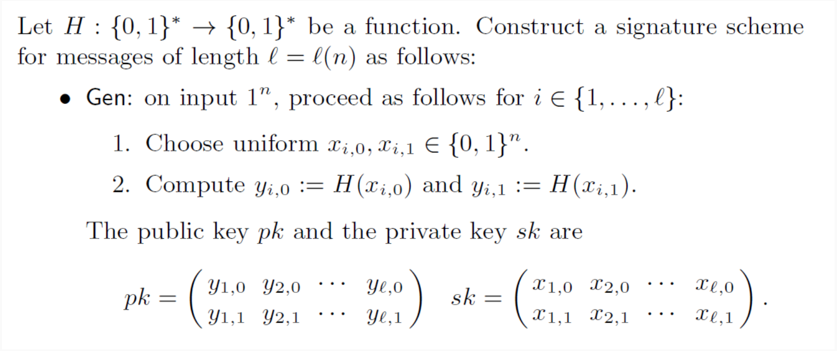
- Hash-Based Signatures
- Authenticated Key-Exchange using Digital Signatures

Hash-Based Signatures

Lamport One-Time Signature (OTS) - ’79

- Very interesting construction, that happens to use **only a hash function**
- **No other mathematical hard problem necessary!**

The construction:



Let $H : \{0,1\}^* \to \{0,1\}^*$ be a function. Construct a signature scheme for messages of length $\ell = \ell(n)$ as follows:

- Gen: on input 1^n, proceed as follows for $i \in \{1,\ldots,\ell\}$:
  1. Choose uniform $x_{i,0}, x_{i,1} \in \{0,1\}^n$.
  2. Compute $y_{i,0} := H(x_{i,0})$ and $y_{i,1} := H(x_{i,1})$.

  The public key pk and the private key sk are

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}.$$

Lamport One-Time Signature (OTS) - ’79

The construction (contd.):

- **Sign:** on input a private key sk as above and a message $m \in \{0,1\}^\ell$ with $m = m_1 \cdots m_\ell$, output the signature $(x_{1,m_1}, \ldots, x_{\ell,m_\ell})$.
- **Vrfy:** on input a public key pk as above, a message $m \in \{0,1\}^\ell$ with $m = m_1 \cdots m_\ell$, and a signature $\sigma = (x_1, \ldots, x_\ell)$, output 1 if and only if $H(x_i) = y_{i,m_i}$ for all $1 \leq i \leq \ell$.

Example:

Signing $m = 011$:

$$sk = \begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & x_{3,0} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0},\, x_{2,1},\, x_{3,1})$$

Verifying for $m = 011$ and $\sigma = (x_1, x_2, x_3)$:

$$pk = \left.\begin{pmatrix} \boxed{y_{1,0}} & y_{2,0} & y_{3,0} \\ y_{1,1} & \boxed{y_{2,1}} & \boxed{y_{3,1}} \end{pmatrix}\right\} \Rightarrow \begin{matrix} H(x_1) \overset{?}{=} y_{1,0} \\ H(x_2) \overset{?}{=} y_{2,1} \\ H(x_3) \overset{?}{=} y_{3,1} \end{matrix}$$

Lamport One-Time Signature (OTS) - '79

- Works **only for one** signature!
- Trivially forgeable given two valid signatures (if different in more than one bit)

Example:

$$\begin{pmatrix} \boxed{x_{1,0}} & x_{2,0} & x_{3,0} \\ x_{1,1} & \boxed{x_{2,1}} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0},\, x_{2,1},\, x_{3,1})$$

$$\begin{pmatrix} x_{1,0} & \boxed{x_{2,0}} & x_{3,0} \\ \boxed{x_{1,1}} & x_{2,1} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,1},\, x_{2,0},\, x_{3,1})$$

"Mix and match" forgery

$$\begin{pmatrix} \boxed{x_{1,0}} & \boxed{x_{2,0}} & x_{3,0} \\ x_{1,1} & x_{2,1} & \boxed{x_{3,1}} \end{pmatrix} \Rightarrow \sigma = (x_{1,0},\, x_{2,0},\, x_{3,1})$$

Security of One-Time Signatures

Existential unforgeability under **single message attack (EUF-SMA)**

A Digital Signature scheme *Dss* is called EUF-SMA-secure (one-time-EUF-CMA-secure) if any PPT algorithm $\mathcal{A}$ has only negligible success probability

$$\mathrm{Succ}^{\text{euf-sma}}_{\mathsf{DSs}(1^k)}(\mathcal{A}) = Pr\left[\mathsf{Exp}^{\text{euf-sma}}_{\mathsf{DSs}(1^k)}(\mathcal{A}) = Accept\right].$$

in the following $\mathsf{Exp}^{\text{euf-sma}}_{\mathsf{DSs}(1^k)}(\mathcal{A})$ **game (experiment)**:

Challenger		**Adversary**
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}()$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{M'}$	single message M'
	$\xrightarrow{\mathsf{Sign}(\mathsf{sk}, M')}$	
	$\xleftarrow{(M^*, \sigma^*)}$	M^*, σ^*
Return 1 iff $\mathsf{Vf}(\mathsf{pk}, M, \sigma) = Accept$ otherwise 0.		

Security of Lamport OTS

If H is one-way function, Lamport OTS is one-time secure.

Idea of proof:

- During the game, $\mathcal{A}$ asks for the signature of M'
- Consider any other message $M \neq M'$
- There must be at least one position $i \in \{1, \ldots, \ell\}$ on which they differ. Let $M_i = b \neq M'_i$.
- Then forging a signature on M requires, at least, finding a preimage (under H) of element $y_{i,b}$ of the public key.
- Since H is one-way, this is infeasible.

Technical hints:

- Inverter $\mathcal{I}$ wants to invert H at given y
- Prepares "appropriate" input public key to $\mathcal{A}$, s.t. $y_{i^*,b^*} = y$ for a randomly chosen i^*, b^*
- The rest of the public key is generated properly, as in the key generation
- Probability of success - $1/2\ell$ times probability of success of $\mathcal{A}$

Merkle Signature Scheme (MSS) - '89

- Merkle extends Lamport's OTS to multi-time signature scheme, using binary tree (Merkle tree)
- Each leaf is public key of OTS
- OTS are used sequentially (must keep track of index! - **stateful** signature scheme)
- Signature is $\sigma = (i, \text{Sign}_{OTS}(M, X_i), Y_i, Auth)$

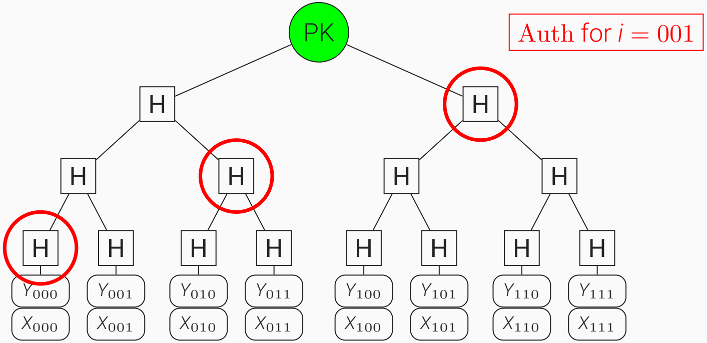


image credit: Peter Schwabe

Merkle Signature Scheme (MSS) - ’89

- Let $H : \{0,1\}^* \rightarrow \{0,1\}^s$ be a cryptographic hash function
- Assume that a one-time signature scheme (OTS) is given
- For $h \in \mathbb{N}$, 2^h signatures are to be generated that are verifiable with one MSS public key.

MSS:

KeyGen:

1. generate 2^h OTSS key pairs (X_i, Y_i), $i \in \{1, ..., 2^h\}$, where the X^i are the signature keys and Y^i the verification keys.
2. The MSS private key is $(X_1, \ldots, X_{2^h})$.
3. To determine the MSS public key, construct a binary authentication tree as follows. The leafs of the authentication tree are the hash values $H(Y_i)$ of the verification keys. Each inner node (including the root) of the tree is the hash value of the concatenation of its two children. The MSS public key is the root of the authentication tree.

Merkle Signature Scheme (MSS) - '89

MSS:

Sign: Given message M,

1. The i-th message is signed with the i-th key par X_i, Y_i.
2. The signature is $\sigma = (i, \text{Sign}_{OTS}(M, X_i), Y_i, Auth)$
3. The authentication path $Auth$ is a sequence of nodes $(a_h, ..., a_1)$ in the authentication tree - the siblings needed to go up in the tree

Verify: To verify the message - signature pair (M, σ)

1. The verifier first verifies the one-time signature with the verification key Y_i. If this verification fails, the verifier rejects the MSS signature as invalid.
2. Otherwise, the verifier constructs a sequence of nodes of the tree of length $h + 1$ starting from the hash $H(Y_i)$ of the verification key Y_i.
    - Each node is constructed as the hash of the previous node and its sibling that is part of the authentication path
3. The verifier accepts if the last node in the sequence is the MSS public key.

Security of MSS

MSS is EUF-CMA secure if OTS is a one-time EUF-CMA secure signature scheme and H is a random element from a family of collision resistant hash functions.

- Suppose an adversary $\mathcal{B}$ exists that breaks the EUF-CMA security of MSS.
- We build an adversary $\mathcal{A}$ that forges an OTS or finds a collision for H
- Procedure:
  1. In the one-time EUF-CMA security game, $\mathcal{A}$ receives a public key Y, which it puts as leaf i_*. It also has access to the one-time signing oracle corresponding to this public key
  2. $\mathcal{A}$ calls the KGen algorithm of OTS, to obtain 2^N pairs $(X_i, Y_i), i \neq i_*$ of private-public keys
  3. It constructs a Merkle tree of depth N, with root R
  4. It runs $\mathcal{B}$ against the Merkle signature, and $\mathcal{B}$ outputs a forgery $(M', \sigma' = (i, \mathrm{Sign}_{OTS}(M', X_i'), Y_i', Auth'))$
  5. In the game, $\mathcal{B}$ asks the signatures of various messages and $\mathcal{A}$ computes genuine signatures for the queried messages σ_i

Security of MSS

MSS is EUF-CMA secure if OTS is a one-time EUF-CMA secure signature scheme and H is a random element from a family of collision resistant hash functions.

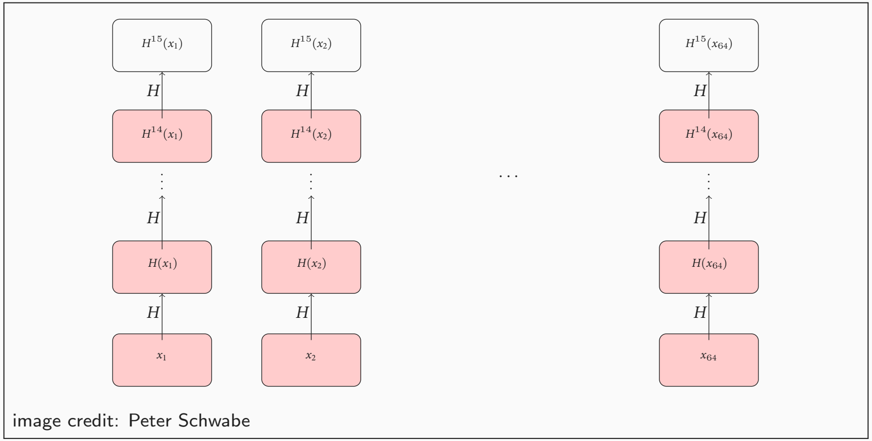
- Procedure (contd.):
  6. After the forgery is output, $\mathcal{A}$ checks whether the number of queries is higher than i. If yes, a message was queried and a genuine signature has been produced for i. If not, it continues to sign the message until it reaches the index i.
  7. The genuine signature is $(M', \sigma = (i, \text{Sign}_{OTS}(M', X_i), Y_i, Auth))$
  8. If $(Y_i, Auth) \neq (Y_i', Auth')$, then a collision has been found, otherwise, $\mathcal{A}$ outputs a forgery $\text{Sign}_{OTS}(M', X_i')$
- Success probability:
  - Let ϵ be the success probability of the forger
  - If ρ - probability that genuine and forged auth. paths are different, collision can be found with prob. $\rho\epsilon$,
  - Otherwise success of $1/2^N(1-\rho)\epsilon$ to have $i = i^*$

Winternitz OTS (WOTS)

- Lamport's scheme produces extremely large signatures
  - for ex. for hash function with 256-bit output, in order to sign 256-bit messages, the keys are of size $2 \times 256 \times 32B = 16KB$ and the signature $256 \times 32B = 8KB$
- Idea - hash chains
- Winternitz OTS

- **Key generation:**
  - Generate 256-bit random values $(x_1, x_2, \ldots, x_{64})$ (secret key)
  - Compute $(y_1, y_2, \ldots, y_{64}) = (H^{15}(x_1), ..., H^{15}(x_{64}))$ (public key)
- **Signing:**
  - Chop 256 bit message into 64 chunks of 4 bits $M = (M_1, ..., M_{64})$
  - Compute $\sigma = (\sigma_1, ..., \sigma_{64}) = (H^{M_1}(x_1), ..., H^{M_{64}}(x_{64}))$
- **Verification:**
  - Check that $y_1 = H^{15-M_1}(\sigma_1), ..., y_{64} = H^{15-M_{64}}(\sigma_{64})$

Winternitz OTS



- Length of chain - Trade-off between shorter signatures and speed
- Trivial attack - from signature of $(\dots, M_i, \dots)$ obtain signature of $(\dots, M_i + 1, \dots)$
    - Solution: two chains - one forward one backward
    - Smarter solutions exist with better efficiency

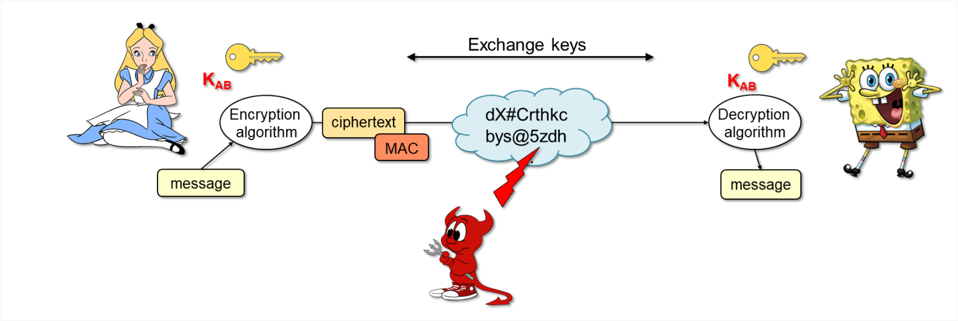
Standardization of statefull signatures

- RFC 8391 – XMSS: eXtended Merkle Sigthenature Scheme
  - protection against multi target attacks
  - tight security
  - keys and signatures - a few kilobytes
  - perfect for applications where it is natural to count signatures - updates, code signing
  - stateful means no reuse of OTS key allowed, or going back to previous state!
  - so not good for back-ups, multithreading, VMs
- RFC 8554 – LMS (Leighton-Micali signature)
- NIST endorses XMSS an LMS
- ISO SC27 JTC1 WG2 - study period on stateful hash signatures

- SPHINCS and SPHINCS+ - stateless hash-based signatures
  - SPHINCS+ - maybe it gets standardize by NIST. . . let's wait a few more weeks
  - maybe it will be included in the syllabus next year!

Authenticated Key-Exchange using Digital Signatures

Recall our everyday scenario



- Alice and Bob have **not** agreed on a joint key yet, but they want to communicate securely
  - They want to exchange symmetric keys over the public channel, first
  - They use public key cryptography for this, so that Eve can't learn the key
  - Typically using (Merkle-)Diffie-Hellman key exchange

Recall the magic of (Merkle-)Diffie-Hellman key exchange

Public-key based establishment of a shared secret

Alice's client		Bob's server
P, G, a		P, G, b
$A \leftarrow G^a$	$\xrightarrow{\text{Alice}; A}$	
	$\xleftarrow{\text{Bob}; B}$	$B \leftarrow G^b$
$K_{A,B} \leftarrow B^a$		$K_{B,A} \leftarrow A^b$

Alice and Bob arrive at the same shared secret $K_{A,B} = K_{B,A}$

$$K_{A,B} = (B^a) = (G^b)^a = G^{b \cdot a} = G^{a \cdot b} = (G^a)^b = A^b = K_{B,A}$$

- Alice and Bob derive key(s) from secret: $K \leftarrow KDF(K_{A,B})$
- They use K further in their communication for encryption or message authentication

Man-in-the-middle attack on (Merkle-)Diffie-Hellman Key Exchange

Alice's client		Eve		Bob's server
P, G, a		P, G, e		P, G, b
$A \leftarrow G^a$	$\xrightarrow{\text{Alice;}A}$	$E \leftarrow G^e$	$\xrightarrow{\text{Alice;}E}$	
			$\xleftarrow{\text{Bob;}B}$	$B \leftarrow G^b$
	$\xleftarrow{\text{Bob;}E}$			
$K_{A,B} \leftarrow E^a$		$K_{A,B} \leftarrow A^e$		
		$K_{B,A} \leftarrow B^e$		$K_{B,A} \leftarrow E^b$

- Alice and Bob both unknowingly share a secret with Eve,
- In subsequent exchange protected with shared secrets
  - **Eve decrypts, can read plaintext, and re-encrypts**
  - Eve may **modify/delete messages and compute tags**
- **Problem: Alice and Bob can never be sure who sent the message**
- **Solution: Entity authentication/Identification**
  - Alice must verify B really comes from Bob and vice versa

Let’s try to fix it

- Alice has a long term signing key pair $(\mathsf{pk}_A, \mathsf{sk}_A)$ and Bob has $(\mathsf{pk}_B, \mathsf{sk}_B)$
- **Very important:**
    - their long term keys are already authenticated!
    - Alice knows pk_B belongs to Bob and Bob knows pk_A belongs to Alice

The fix:

Alice’s client		Bob’s server
$P, G, \mathsf{pk}_B, a, \mathsf{sk}_A$		$P, G, \mathsf{pk}_A, b, \mathsf{sk}_B$
$A \leftarrow G^a$	$\xrightarrow{\text{Alice; } A}$	
$\mathsf{Vf}_{\mathsf{pk}_B}(\sigma_B)$	$\xleftarrow{\text{Bob; } B;\ \sigma_B}$	$B \leftarrow G^b,\ \sigma_B = \mathsf{Sign}_{\mathsf{sk}_B}(A, B)$
$\sigma_A = \mathsf{Sign}_{\mathsf{sk}_A}(B, A)$	$\xrightarrow{\sigma_A}$	$\mathsf{Vf}_{\mathsf{pk}_A}(\sigma_A)$
$K_{A,B} \leftarrow B^a$		$K_{B,A} \leftarrow A^b$

- “Authentication-only” Station-to-Station Protocol (Diffie et al. ’92)
    - **claimed mutual authentication!**
- Flawed! Man-in-the-middle attacks possible!

Man-in-the-middle attack on the “Auth.-only” STS protocol – Attack 1

Alice’s client		Eve		Bob’s server
$P, G, \mathsf{pk}_B, \mathsf{pk}_E,$ a, sk_A		$P, G, \mathsf{pk}_A, \mathsf{pk}_B,$ e, sk_E		$P, G, \mathsf{pk}_A, \mathsf{pk}_E,$ b, sk_B
$A \leftarrow G^a$	$\xrightarrow{\text{Alice; } A}$		$\xrightarrow{\text{Alice; } A}$	
				$B \leftarrow G^b$
$\mathsf{Vf}_{\mathsf{pk}_E}(\sigma_E)$	$\xleftarrow{\text{Eve; } B;\ \sigma_E}$	$\sigma_E = \mathsf{Sign}_{\mathsf{sk}_E}(A, B)$	$\xleftarrow{\text{Bob; } B;\ \sigma_B}$	$\sigma_B = \mathsf{Sign}_{\mathsf{sk}_B}(A, B)$
$\sigma_A = \mathsf{Sign}_{\mathsf{sk}_A}(B, A)$	$\xrightarrow{\sigma_A}$		$\xrightarrow{\sigma_A}$	$\mathsf{Vf}_{\mathsf{pk}_A}(\sigma_A)$
$K_{A,B} \leftarrow B^a$				$K_{B,A} \leftarrow A^b$

- Alice thinks she is talking to Eve,
- Bob thinks he is talking to Alice
- So, no mutual authentication!, but Eve does not know the key
- **Identity misbinding attack**
    - but Eve can trick Alice into saying things not intended for Bob, but for Eve
    - and then just relay them to Bob
    - Think of: I agree to buy your house for 1/2 million (Eve has a mansion, and Bob a shed)

Man-in-the-middle attack on the “Auth.-only” STS protocol – Attack 2

Alice’s client		Eve		Bob’s server
$P, G, \mathsf{pk}_B, \mathsf{pk}_E,$ a, sk_A		$P, G, \mathsf{pk}_A, \mathsf{pk}_B,$ e, sk_E		$P, G, \mathsf{pk}_A, \mathsf{pk}_E,$ b, sk_B
$A \leftarrow G^a$	$\xrightarrow{\text{Alice; } A}$		$\xrightarrow{\text{Eve; } A}$	
				$B \leftarrow G^b$
$\mathsf{Vf}_{\mathsf{pk}_B}(\sigma_B)$	$\xleftarrow{\text{Bob; } B;\ \sigma_B}$		$\xleftarrow{\text{Bob; } B;\ \sigma_B}$	$\sigma_B = \mathsf{Sign}_{\mathsf{sk}_B}(A, B)$
$\sigma_A = \mathsf{Sign}_{\mathsf{sk}_A}(B, A)$	$\xrightarrow{\sigma_A}$	$\sigma_E = \mathsf{Sign}_{\mathsf{sk}_E}(B, A)$	$\xrightarrow{\sigma_E}$	$\mathsf{Vf}_{\mathsf{pk}_E}(\sigma_E)$
$K_{A,B} \leftarrow B^a$				$K_{B,A} \leftarrow A^b$

- Alice thinks she is talking to Bob,
- Bob thinks he is talking to Eve
- So, no mutual authentication!, but Eve does not know the key
- Again, **Identity misbinding attack**
- but Bob thinks everything from Alice comes from Eve
- Similar effect as previously, but now Eve intercepts the initial message from Alice intended for Bob
- in Attack 1, Eve uses a legitimate initial message from Alice

A fixed protocol – ISO-9796

Alice's client		Bob's server
$P, G, \mathrm{pk}_B, a, \mathrm{sk}_A$		$P, G, \mathrm{pk}_A, b, \mathrm{sk}_B$
$A \leftarrow G^a$	$\xrightarrow{\text{Alice};\ A}$	
$\mathrm{Vf}_{\mathrm{pk}_B}(\sigma_B)$	$\xleftarrow{\text{Bob};\ B;\ \sigma_B}$	$B \leftarrow G^b,\ \sigma_B = \mathrm{Sign}_{\mathrm{sk}_B}(A, B, \mathit{Alice})$
$\sigma_A = \mathrm{Sign}_{\mathrm{sk}_A}(B, A, \mathit{Bob})$	$\xrightarrow{\sigma_A}$	$\mathrm{Vf}_{\mathrm{pk}_A}(\sigma_A)$
$K_{A,B} \leftarrow B^a$		$K_{B,A} \leftarrow A^b$

- Proven secure [Canetti-Krawzyk '01]
- **Include the identity of the receiver in the signature**
- The identities of Alice and Bob are bound to the key $K_{A,B}$ and the previous attacks don't work
  - Eve can't just relay the message sent by Bob (Attack 2)
  - Bob; B; $\mathrm{Sign}_{\mathrm{sk}_B}(A, B, \mathit{Eve})$ needs to be Bob; B; $\mathrm{Sign}_{\mathrm{sk}_B}(A, B, \mathit{Alice})$
  - Where does Attack 1 fail?
- Does including the identity of the sender accomplish the same? (See homework :))

Issues with ISO-9796

- Proven secure but...
- **Neither initiator nor responder privacy protection**: both Alice and Bob need the identity of the peer before being able to proceed with the protocol
    - For ex, roaming users/browsers need initiator identity protection
    - Or NFC cards need responder identity protection
    - Can't be fixed for active attackers (as are Attacks 1 and 2), as the identity of the peer must be known before authentication (to be included in the signature)
- **Non-repudiability**: by signing the identity of the peer, one leaves a non-deniable proof of communication with that peer
- How to fix these issues?
    - Can we have identity protection for both peers, or even at least for one of them?
    - Major issue in many protocols today (next lectures)

SIGMA (SIGn and MAc) protocols [Hugo Krawczyk '03]

Alice's client		Bob's server
$P, G, \mathrm{pk}_B, a, \mathrm{sk}_A$		$P, G, \mathrm{pk}_A, b, \mathrm{sk}_B$
$A \leftarrow G^a$	$\xrightarrow{\cancel{\text{Alice;}}\ A}$	$B \leftarrow G^b,\ K_{B,A} \leftarrow A^b$
		$K = KDF(K_{B,A}),\ t_B = MAC_K(Bob)$
$\mathrm{Vf}_{\mathrm{pk}_B}(\sigma_B),\ K_{A,B} \leftarrow B^a$	$\xleftarrow{\text{Bob};\ B;\ \sigma_B;\ t_B}$	$\sigma_B = \mathrm{Sign}_{\mathrm{sk}_B}(A, B, \cancel{Alice})$
$K = KDF(K_{A,B}),\ t_A = MAC_K(Alice)$		
$\sigma_A = \mathrm{Sign}_{\mathrm{sk}_A}(B, A, \cancel{Bob})$	$\xrightarrow{\sigma_A;\ t_A}$	$\mathrm{Vf}_{\mathrm{pk}_A}(\sigma_A)$

- We keep the signing as in the STS protocol, since it prevents MiM attacks
- But, remove the identities from the signature, to provide repudiability
- And Alice does not advertize her identity
  - **This is basically STS, so misbinding possible!**
- Solution?:
  - **Alice and Bob MAC their own identity!**
  - From the MAC, Alice can see that Bob's identity was not replaced by Eve's

SIGMA-I protocol (initiator identity protection)

Alice's client		Bob's server
$P, G, \mathrm{pk}_B, a, \mathrm{sk}_A$		$P, G, \mathrm{pk}_A, b, \mathrm{sk}_B$
$A \leftarrow G^a$	$\xrightarrow{A}$	$B \leftarrow G^b,\ K_{B,A} \leftarrow A^b,\ K_M; K_E = KDF(K_{B,A})$
		$t_B = MAC_{K_M}(Bob),\ \sigma_B = \mathrm{Sign}_{\mathrm{sk}_B}(A, B)$
$K_{A,B} \leftarrow B^a,\ K_M; K_E = KDF(K_{A,B}),$	$\xleftarrow{B;\ C_B}$	$C_B = Enc_{K_E}(Bob, \sigma_B, t_B)$
$t_A = MAC_{K_M}(Alice),\ \sigma_A = \mathrm{Sign}_{\mathrm{sk}_A}(B, A)$		
$C_A = Enc_{K_E}(Alice, \sigma_A, t_A)$	$\xrightarrow{C_A}$	

*For compactness, signature and MAC verification, decryption steps omitted (but are performed)

- No identities in clear $\Rightarrow$ protection against passive attackers
- Alice can verify the identity of Bob before disclosing her own identity $\Rightarrow$ **identity protection of initiator** against active attackers
- Signing $\Rightarrow$ MiM attacks prevented
- MAC of own identity $\Rightarrow$ identity misbinding attacks prevented
- **To be used in TLS 1.3**

SIGMA-R protocol (responder identity protection)

Alice's client		Bob's server
$P, G, \mathrm{pk}_B, a, \mathrm{sk}_A$		$P, G, \mathrm{pk}_A, b, \mathrm{sk}_B$
$A \leftarrow G^a$	$\xrightarrow{A}$	$B \leftarrow G^b$
$K_{A,B} \leftarrow B^a,\ K_M; K_E = KDF(K_{A,B}),$	$\xleftarrow{B}$	
$t_A = MAC_{K_M}(Alice),\ \sigma_A = \mathrm{Sign}_{\mathrm{sk}_A}(B, A)$		
$C_A = Enc_{K_E}(Alice, \sigma_A, t_A)$	$\xrightarrow{C_A}$	$K_{B,A} \leftarrow A^b,\ K'_M; K'_E = KDF(K_{B,A})$
		$t_B = MAC_{K'_M}(Bob),\ \sigma_B = \mathrm{Sign}_{\mathrm{sk}_B}(A, B)$
	$\xleftarrow{C_B}$	$C_B = Enc_{K'_E}(Bob, \sigma_B, t_B)$

*For compactness, signature and MAC verification, decryption steps omitted (but are performed)

- Bob can verify the identity of Bob before disclosing his own identity $\Rightarrow$ **identity protection of responder** against active attackers
- Alice and Bob use different $K_M; K_E$ and $K'_M; K'_E$ to prevent **reflection attacks**
  - Hint: Protocol is completely symmetric!

SIGMA protocol in IKE (Internet Key Exchange) main mode

$$
\begin{array}{lcl}
\text{Alice's client} & & \text{Bob's server} \\
\hline
P, G, \mathrm{pk}_B, a, \mathrm{sk}_A & & P, G, \mathrm{pk}_A, b, \mathrm{sk}_B \\
\hline
A \leftarrow G^a & \xrightarrow{A} & B \leftarrow G^b \\
K_{A,B} \leftarrow B^a,\ K_M; K_E = KDF(K_{A,B}), & \xleftarrow{B} & \\
t_A = MAC_{K_M}(Alice),\ \sigma_A = \mathrm{Sign}_{\mathrm{sk}_A}(B, A, t_A) & & \\
C_A = Enc_{K_E}(Alice, \sigma_A, \cancel{t_A}) & \xrightarrow{C_A} & K_{B,A} \leftarrow A^b,\ K'_M; K'_E = KDF(K_{B,A}) \\
 & & t_B = MAC_{K'_M}(Bob),\ \sigma_B = \mathrm{Sign}_{\mathrm{sk}_B}(A, B, t_B) \\
 & \xleftarrow{C_B} & C_B = Enc_{K'_E}(Bob, \sigma_B, \cancel{t_B}) \\
\hline
\end{array}
$$

*For compactness, signature and MAC verification, decryption steps omitted (but are performed)

**Difference from SIGMA-R in green

- IKE is core AKE protocol of IPSec – IP Security [RFC2401-12]
- above is IKE v1
- Aggressive mode exists without identity protection
- IKE v2 slight differences – A, B not included inside of MAC, only identity

Summary

Today:

- Hash-Based signatures
- Authenticated Key-Exchange using Digital Signatures

Next time:

- **“Challenges of Post-Quantum Cryptography in the Embedded World”** Christine van Vredendaal, NXP
- **“Secure messaging beyond content itself”** - Sofía Celi, Brave