> Display --full title

>>> Threshold Signatures

>>> with Offline Parties

Name: Alessio Meneghetti[†]

Date: May 27, 2021 @ CrypTO Conference 2021

[†] alessio.meneghetti@unitn.it

>>> ls -d */

1. Digital Signatures
   ECDSA

2. Threshold Signatures
   $(2,3)$-Threshold ECDSA

3. Threshold Signatures with Offline Participants
   $(2,3)$-Threshold ECDSA with an offline participant

4. A security result

>>> Digital Signatures

Definition

A cryptographic primitive acting as a digital counterpart of a handwritten signature

Properties

> Non-repudiation
> Authentication
> Integrity
> Unforgeability

>>> Digital Signatures

Key-Generation Algorithm (Alice)

> input: $\emptyset$

> output: private key sk public key pk

Signing Algorithm (Alice)

> input: a message M, a private key sk

> output: a signature σ of the message M

Verification Algorithm (Bob)

> input: a message M, a signature σ, a public key pk

> output: True or False

>>> ECDSA

~~Elliptic curves in short Weierstrass Form: $y^2 = x^3 + ax + b$ over a field $\mathbb{F}_p$ of prime order p.~~

~~The rational points are the pairs (x, y) of elements of $\mathbb{F}_p$ satisfying the equation, together with one extra point at infinity $\mathcal{O}$.~~

Elliptic curves

A group $(E, +)$ of prime order q generated by a point $B = (B_x, B_y)$ such that the DLOG $Q = uB$ is hard to solve

ECDSA Parameters

> a base point B of E with prime order q

> a Hash function H

>>> ECDSA: Key-Generation

Key-Generation (Alice)

- Input:
  - $\emptyset$
- Procedure:
  - Pick an integer u at random in the interval $[1, q-1]$.
  - Compute the point $\mathcal{Q} = uB$.
- Output:
  - the key-pair $sk = u$, $pk = \mathcal{Q}$.

>>> ECDSA: Signature Algorithm

Signing (Alice)

- Input:
  - a key-pair $(u, \mathcal{Q})$
  - a message digest $\mathrm{H}(M)$
- Procedure:
  - Pick an integer k at random in the interval $[1, q-1]$.
  - Compute the point $\mathcal{R} = k^{-1}B$.
  - Compute $s = k(\mathrm{H}(M) + ru)$ with $r = \mathcal{R}_x$.
- Output:
  - the signature (r, s).

>>> ECDSA: Verification Algorithm

Verification (Bob)

- Input:
  - a message M
  - a signature (r, s) of M
  - a public key $\mathcal{Q}$
- Procedure:
  * Compute $c_1 = \mathrm{H}(M)s^{-1}$ and $c_2 = rs^{-1}$,
  * Compute the point $\mathcal{C} = c_1 B + c_2 \mathcal{Q}$,
- Output:
  - True if $r = \mathcal{C}_x$, False otherwise

>>> Threshold Signatures

Definition ((t, n)-Threshold Signatures)

Just like a standard digital signature, except that

- Alice is replaced by a group of n players
- At least t among them have to agree in order to sign a document
- The Key-Generation is a multi-party protocol involving all n players
- The Signature Algorithms is a multi-party protocol involving at least t players

Remark

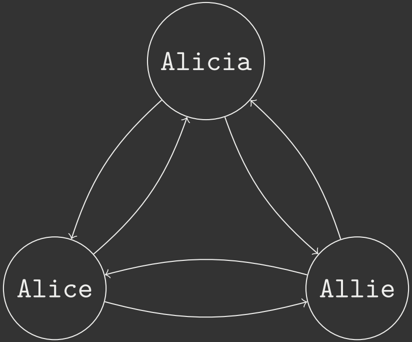
The verification algorithm is the same as the one of a "standard" digital signature

>>> Threshold Signatures

- 1995: S. Langford:
  Threshold DSS signatures without a trusted party
- 1996: R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin:
  Robust threshold DSS signatures
- Boneh, Canetti, Doerner, Kondi, Lee, Lindell, MacKenzie, Magri, Makriyannis, Narayanan, Nof, Orlandi, Peled, Reiter, Shelat, Shlomovits, ...
- 2018: R. Gennaro and S. Goldfeder:
  Fast multiparty threshold ECDSA with fast trustless setup
- 2021: M. Battagliola, R. Longo, A. Meneghetti, M. Sala,
  Threshold ECDSA with an Offline Recovery Party

>>> $(2,3)$-Threshold ECDSA

> Alice, Allie, Alicia share the right to sign together



> Bob uses the verification algorithm of ECDSA

>>> How to share a secret

Many protocols implement (some sort of) Shamir's scheme to share a secret

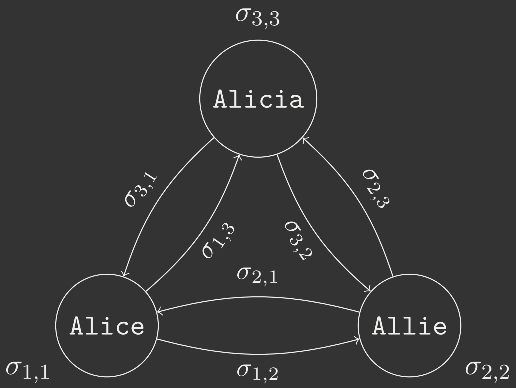
Shamir's idea

- a Dealer chooses a polynomial f, such that $\sigma = f(0) \in \mathbb{Z}_q$ is the secret to be shared:

$$f(x) = \sigma + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$$

- the Dealer sends to each player P_i the value $\sigma_i = f(i) \in \mathbb{Z}_q$

>>> $(2,3)$-Threshold ECDSA: sharing shards



>>> $(2,3)$-Threshold ECDSA: Key-Generation

Alice	Allie & Alicia
Randomly chooses:	
$u_1, m_1 \in \mathbb{Z}_q$	
Computes:	
$u_1 B$	$\rightarrow$ Allie, Alicia
$f_1(X) = u_1 + m_1 X$	
$\sigma_{1,1} = f_1(2)$	
$\sigma_{1,2} = f_1(3)$	$\rightarrow$ Allie
$\sigma_{1,3} = f_1(1)$	$\rightarrow$ Alicia
$x_1 = \sigma_{1,1} + \sigma_{2,1} + \sigma_{3,1}$	
private key:	
$\omega_1 = t \cdot x_1$	

>>> $(2,3)$-Threshold ECDSA: Keys

- the public key is $Q = u_1 B + u_2 B + u_3 B = (u_1 + u_2 + u_3)B$
- the "global" private key $u = u_1 + u_2 + u_3$ is unknown to anyone
- the coefficients for the private keys depend on the set of active parties:
  - if Alice and Allie want to sign,

$$\omega_1 = 3x_1, \qquad \omega_2 = -2x_2$$

  - if Alice and Alicia want to sign,

$$\omega_1 = -x_1, \qquad \omega_3 = 2x_3$$

  - if Allie and Alicia want to sign,

$$\omega_2 = -\frac{1}{2}x_2, \qquad \omega_3 = \frac{3}{2}x_3$$

>>> $(2,3)$-Threshold ECDSA: Keys

Example: Alice and Allie

Alice's private key: $\omega_1 = 3x_1$

Allie's private key: $\omega_2 = -2x_2$

Suppose they are able to sum their own private keys:

$$\begin{aligned}
\omega_1 + \omega_2 &= 3x_1 - 2x_2 \\
&= 3(\sigma_{1,1} + \sigma_{2,1} + \sigma_{3,1}) - 2(\sigma_{1,2} + \sigma_{2,2} + \sigma_{3,2}) \\
&= 3(f_1(2) + f_2(2) + f_3(2)) - 2(f_1(3) + f_2(3) + f_3(3)) \\
&= 3(u_1 + m_1 \cdot 2 + u_2 + m_2 \cdot 2 + u_3 + m_3 \cdot 2) \\
&\qquad -2(u_1 + m_1 \cdot 3 + u_2 + m_2 \cdot 3 + u_3 + m_3 \cdot 3) \\
&= u_1 + u_2 + u_3 = u
\end{aligned}$$

Recall

The "global" private key is u and the public key is $Q = uB$

>>> $(2,3)$-Threshold ECDSA: Signature

Alice and Allie have to compute together an ECDSA signature of M, i.e. (r,s) where

$$s = k(\mathrm{H}(M) + ru) = k\mathrm{H}(M) + (ku)r$$

It is possible if

- both know r
- Alice knows the additive shard k_1 of $k = k_1 + k_2$
- Allie knows the additive shard k_2 of $k = k_1 + k_2$
- Alice knows the additive shard σ_1 of $ku = \sigma_1 + \sigma_2$
- Allie knows the additive shard σ_2 of $ku = \sigma_1 + \sigma_2$

In this way

- Alice computes $s_1 = k_1\mathrm{H}(M) + \sigma_1 r$
- Allie computes $s_2 = k_2\mathrm{H}(M) + \sigma_2 r$
- $s = s_1 + s_2 = k\mathrm{H}(M) + (ku)r$

>>> $(2, 3)$-Threshold ECDSA: how?

What Alice and Allie really know:

Alice	Allie
a random k_1	a random k_2
her private key ω_1	her private key ω_2

Remark

$$\omega_1 + \omega_2 = u$$

hence

$$ku = (k_1 + k_2)(\omega_1 + \omega_2)$$

Problem

How to obtain additive shards of ku knowing additive shards of k and u?

>>> Multiplicative to Additive conversion (MtA)

Setting

> Alice knows a secret $a_1 \in \mathbb{Z}_q$
> Allie knows a secret $a_2 \in \mathbb{Z}_q$
> we think of a_1 and a_2 as multiplicative shares of a secret $x = a_1 a_2 \mod q$

- Alice knows a secret $a_1 \in \mathbb{Z}_q$
- Allie knows a secret $a_2 \in \mathbb{Z}_q$
- we think of a_1 and a_2 as multiplicative shares of a secret $x = a_1 a_2 \mod q$

Result

- Alice obtains an additive secret share $\alpha_1 \in \mathbb{Z}_q$
- Allie obtains an additive secret share $\alpha_2 \in \mathbb{Z}_q$
- $\alpha_1 + \alpha_2 = x \mod q$

Remark

This can be achieved by using partially-homomorphic encryption schemes, such as the Paillier cryptosystem

>>> $(2,3)$-Threshold ECDSA: additive shards of ku

First MtA step

	Alice	Allie
Input	k_1	ω_2
Output	$\mu_{1,2}$	$\nu_{1,2}$

Recall: $\mu_{1,2} + \nu_{1,2} = k_1\omega_2$

Second MtA step

	Alice	Allie
Input	ω_1	k_2
Output	$\nu_{2,1}$	$\mu_{2,1}$

Recall: $\mu_{2,1} + \nu_{2,1} = k_2\omega_1$

Final step

	Alice	Allie
Input	$k_1, \omega_1, \mu_{1,2}, \nu_{2,1}$	$k_2, \omega_2, \mu_{2,1}, \nu_{1,2}$
Output	$\sigma_1 = k_1\omega_1 + \mu_{1,2} + \nu_{2,1}$	$\sigma_2 = k_2\omega_2 + \mu_{2,1} + \nu_{1,2}$

>>> $(2,3)$-Threshold ECDSA: additive shards of ku

	Alice	Allie
Input	$k_1, \omega_1, \mu_{1,2}, \nu_{2,1}$	$k_2, \omega_2, \mu_{2,1}, \nu_{1,2}$
Output	$\sigma_1 = k_1\omega_1 + \mu_{1,2} + \nu_{2,1}$	$\sigma_2 = k_2\omega_2 + \mu_{2,1} + \nu_{1,2}$

Proof

$$\begin{aligned}
\sigma_1 + \sigma_2 &= (k_1\omega_1 + \mu_{1,2} + \nu_{2,1}) + (k_2\omega_2 + \mu_{2,1} + \nu_{1,2}) \\
&= k_1\omega_1 + (\mu_{1,2} + \nu_{1,2}) + (\mu_{2,1} + \nu_{2,1}) + k_2\omega_2 \\
&= k_1\omega_1 + (k_1\omega_2) + (k_2\omega_1) + k_2\omega_2 \\
&= (k_1 + k_2)(\omega_1 + \omega_2) \\
&= ku
\end{aligned}$$

>>> $(2,3)$-Threshold ECDSA: main idea

Alice and Allie have to compute together an ECDSA signature of M, i.e. (r, s) where

$$s = k(\mathrm{H}(M) + ru) = k\mathrm{H}(M) + (ku)r$$

It is possible if

- both know r
- Alice knows the additive shard k_1 of $k = k_1 + k_2$ ✓
- Allie knows the additive shard k_2 of $k = k_1 + k_2$ ✓
- Alice knows the additive shard σ_1 of $ku = \sigma_1 + \sigma_2$ ✓
- Allie knows the additive shard σ_2 of $ku = \sigma_1 + \sigma_2$ ✓

In this way

- Alice computes $s_1 = k_1\mathrm{H}(M) + \sigma_1 r$
- Allie computes $s_2 = k_2\mathrm{H}(M) + \sigma_2 r$
- $s = s_1 + s_2 = k\mathrm{H}(M) + (ku)r$

>>> $(2,3)$-Threshold ECDSA: main idea

Alice and Allie have to compute together an ECDSA signature of M, i.e. (r, s) where

$$s = k(\mathrm{H}(M) + ru) = k\mathrm{H}(M) + (ku)r$$

It is possible if

> both know r $\checkmark$
> Alice knows the additive shard k_1 of $k = k_1 + k_2$ $\checkmark$
> Allie knows the additive shard k_2 of $k = k_1 + k_2$ $\checkmark$
> Alice knows the additive shard σ_1 of $ku = \sigma_1 + \sigma_2$ $\checkmark$
> Allie knows the additive shard σ_2 of $ku = \sigma_1 + \sigma_2$ $\checkmark$

In this way

> Alice computes $s_1 = k_1\mathrm{H}(M) + \sigma_1 r$
> Allie computes $s_2 = k_2\mathrm{H}(M) + \sigma_2 r$
> $s = s_1 + s_2 = k\mathrm{H}(M) + (ku)r$

>>> Threshold Signatures with offline participants

Definition (Threshold Signatures with offline participants)

Just like a standard (t, n)-threshold digital signature, except that

- Only t out the n parties participate in the key-generation phase
- At least t out of the n parties have to agree in order to sign a document

>>> $(2, 3)$-Threshold ECDSA with an offline participant

- Three parties share the right to sign: Alice, Alicia, Allie
- Online Parties: only Alice and Allie are actively involved in Key-Generation and Signature phases
- Offline party: Alicia goes back online and participates if and only when Alice (or Allie) are incapacitated
- ECDSA-compatibility: Bob uses the verification algorithm of ECDSA

Remark

key-point: Alicia does not want to participate even in the key-generation phase

>>> $(2,3)$-Threshold ECDSA without Alicia: setup

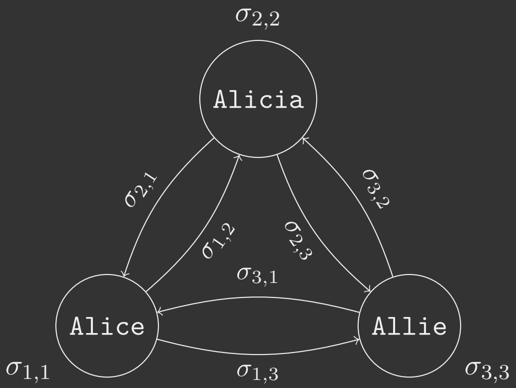
Alice and Allie

- > Secure hash function H
- > Elliptic curve E with group of points of prime order q
- > A generator $B \in E$

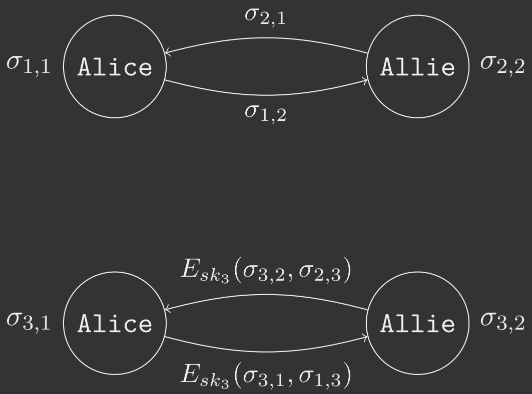
Alicia

- > A key-pair (sk_3, pk_3) for a public-key cipher

>>> Recall: $(2,3)$-Threshold ECDSA Key-Generation



>>> $(2,3)$-Threshold ECDSA Key-Generation without Alicia



>>> $(2,3)$-Threshold ECDSA Signature without Allie

If Allie cannot participate

- Alice contacts Alicia
- Alice sends $E_{sk_3}(\sigma_{3,1}, \sigma_{1,3})$ and $E_{sk_3}(\sigma_{3,2}, \sigma_{2,3})$ to Alicia
- Alicia recover $\sigma_{3,1}, \sigma_{1,3}, \sigma_{3,2}, \sigma_{2,3}$
- Alicia computes u_3 starting from $\sigma_{3,1}, \sigma_{3,2}$
- Alicia generates $\sigma_{3,3}$
- Alicia computes $x_3 = \sigma_{1,3} + \sigma_{2,3} + \sigma_{3,3}$

Finally, Alice and Alicia perform the signature algorithm by using their private keys

$$\omega_1 = -x_1, \qquad \omega_3 = 2x_3$$

>>> $(2,3)$-Threshold ECDSA with an offline participant

Definition (Unforgeability)

A (t,n)-threshold signature scheme is unforgeable if no malicious adversary who corrupts at most $t-1$ players can produce with non-negligible probability the signature on a new message m, given the view of Threshold-Sign on input messages $m_1, \ldots, m_k$ (which the adversary adaptively chooses), as well as the signatures on those messages.

>>> $(2,3)$-Threshold ECDSA with an offline participant

Decisional Diffie-Hellman (DDH) Assumption

Let

* $\mathbb{G}$ be a cyclic group with generator g and order n
* a, b, c be random elements of $\mathbb{Z}_n$

Then, no efficient algorithm can distinguish between the two distributions

$$(g, g^a, g^b, g^{ab}) \qquad \text{and} \qquad (g, g^a, g^b, g^c)$$

>>> $(2, 3)$-Threshold ECDSA with an offline participant

Strong RSA Assumption

Let

* $N = pq$ with both p, q safe primes
* s be a random element of $\mathbb{Z}_N^*$

Then, no efficient algorithm can find

$$x,\ e \neq 1 \qquad \text{such that} \qquad x^e = s \mod N$$

>>> $(2, 3)$-Threshold ECDSA with an offline participant

Theorem

Under the following hypotheses:

- ECDSA is unforgeable
- the strong RSA assumption holds
- the DDH assumption holds
- some other technical assumptions

then Threshold ECDSA protocol is unforgeable

>>> Future works and open problems

* (t, n)-Threshold ECDSA with $n - t$ offline participants
* More security analyses
* Compatibility with other known Digital Signatures

> exit

>>> Thank you!