Integrity, Authentication and Confidentiality in Public-Key Cryptography

Houda Ferradi

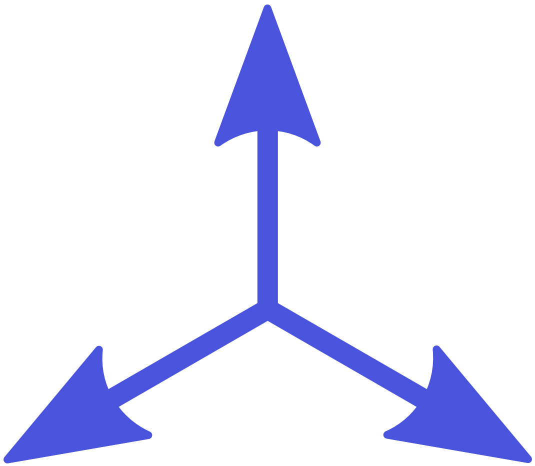
École normale supérieure and PSL Research University

Supported by ANR Project ANR-12-INSE-0014 SIMPATIC

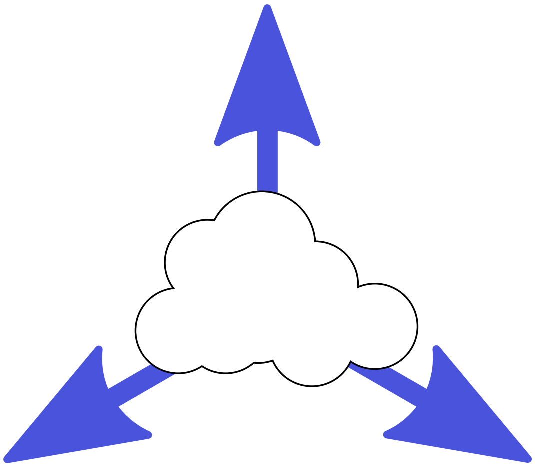
PhD Defense

Thursday, September 22^{nd}, 2016

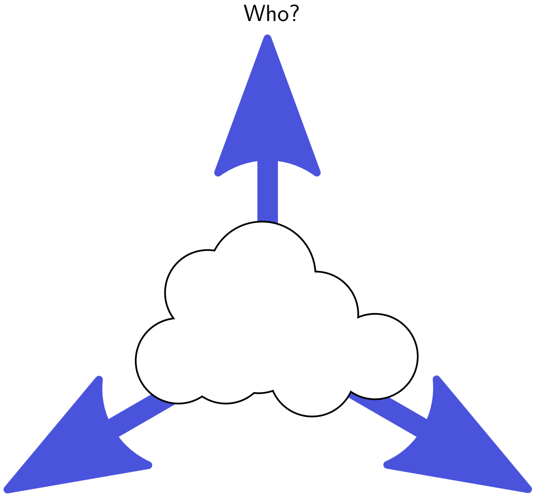
Three Human Concerns



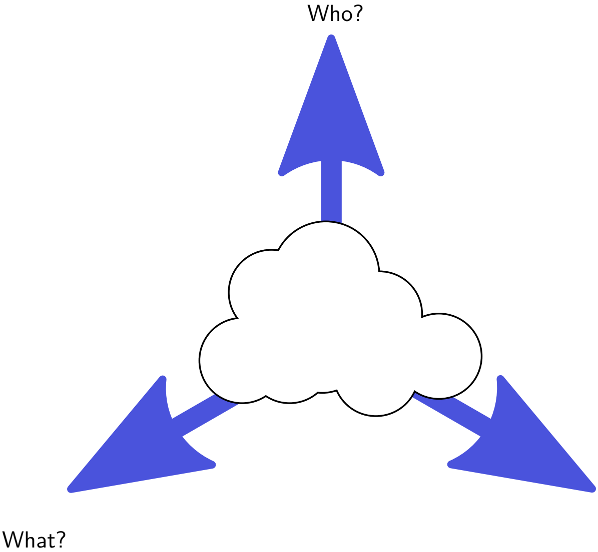
That Become More Acute in the Internet Era



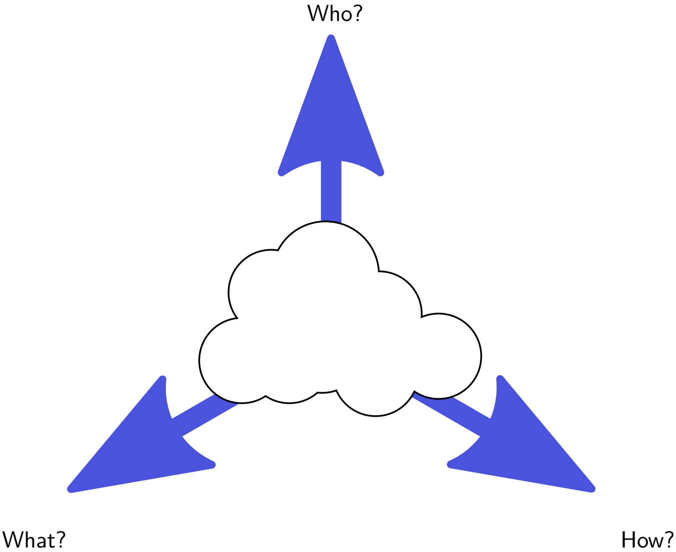
Who Am I Talking To?



What is the Data I Got?

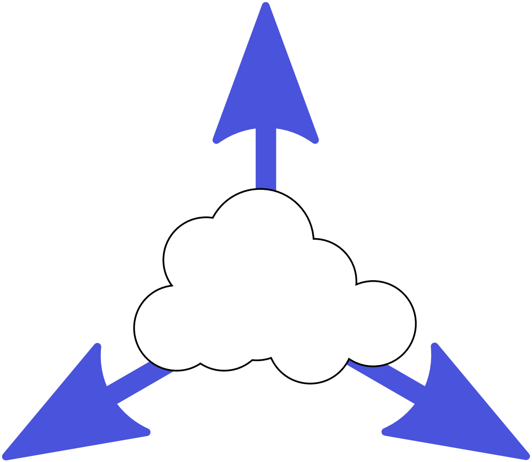


How Was That Data Computed?



Cryptography Answers These Questions

Who? Authentication

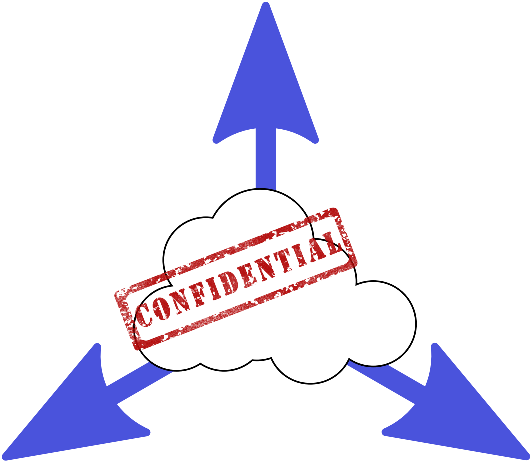


What? Signature

How? Attestation

And Also Provides Confidentiality

Who? Authentication



What? Signature

How? Attestation

Our Agenda

1. Introduction
   - Who am I talking to? [Authentication]
   - What is the data I got? [Digital Signatures]
   - How was that data computed? [Attestation]

Our Agenda

1. Introduction
   - Who am I talking to? [Authentication]
   - What is the data I got? [Digital Signatures]
   - How was that data computed? [Attestation]
2. Three results in these areas
   - Non-interactive modulus attestation
     [For any prime generation algorithm]
   - Legally fair contract signing
     [Allows Bob to prove that Alice behaved unfairly]
   - Thrifty zero-knowledge
     [Increases the efficiency of ZKPs using linear programming]

Our Agenda

1. Introduction
   - Who am I talking to? [Authentication]
   - What is the data I got? [Digital Signatures]
   - How was that data computed? [Attestation]
2. Three results in these areas
   - Non-interactive modulus attestation
     [For any prime generation algorithm]
   - Legally fair contract signing
     [Allows Bob to prove that Alice behaved unfairly]
   - Thrifty zero-knowledge
     [Increases the efficiency of ZKPs using linear programming]
3. Other publications
4. Conclusion

Authentication: Who Am I Talking To?

- Zero-Knowledge Proofs: Goldwasser-Micali-Rackoff [GMR85].

Authentication: Who Am I Talking To?

- Zero-Knowledge Proofs: Goldwasser-Micali-Rackoff [GMR85].
- Reveal nothing but the fact that the prover knows the secret.

Authentication: Who Am I Talking To?

- Zero-Knowledge Proofs: Goldwasser-Micali-Rackoff [GMR85].
- Reveal nothing but the fact that the prover knows the secret.
- Usually implemented as Σ-protocols:

$$
\mathcal{P} \quad \begin{array}{c} \xrightarrow{\quad x \quad} \\ \xleftarrow{\quad c \quad} \\ \xrightarrow{\quad y \quad} \end{array} \quad \mathcal{V}
$$

Authentication: Who Am I Talking To?

- Zero-Knowledge Proofs: Goldwasser-Micali-Rackoff [GMR85].
- Reveal nothing but the fact that the prover knows the secret.
- Usually implemented as Σ-protocols:

$$\mathcal{P} \quad \begin{array}{c} \xrightarrow{\quad x \quad} \\ \xleftarrow{\quad c \quad} \\ \xrightarrow{\quad y \quad} \end{array} \quad \mathcal{V}$$

  - The prover sends a *commitment* x to the verifier;

Authentication: Who Am I Talking To?

- Zero-Knowledge Proofs: Goldwasser-Micali-Rackoff [GMR85].
- Reveal nothing but the fact that the prover knows the secret.
- Usually implemented as Σ-protocols:

$$\mathcal{P} \quad \begin{array}{c} \xrightarrow{\quad x \quad} \\ \xleftarrow{\quad c \quad} \\ \xrightarrow{\quad y \quad} \end{array} \quad \mathcal{V}$$

  - The prover sends a *commitment* x to the verifier;
  - The verifier replies with a *challenge* c;

Authentication: Who Am I Talking To?

- Zero-Knowledge Proofs: Goldwasser-Micali-Rackoff [GMR85].
- Reveal nothing but the fact that the prover knows the secret.
- Usually implemented as Σ-protocols:

$$\mathcal{P} \qquad \begin{array}{c} \xrightarrow{\quad x \quad} \\ \xleftarrow{\quad c \quad} \\ \xrightarrow{\quad y \quad} \end{array} \qquad \mathcal{V}$$

  - The prover sends a *commitment* x to the verifier;
  - The verifier replies with a *challenge* c;
  - The prover gives a *response* y.

Digital Signatures: What is the Data I Got?

Just as handwritten signatures, digital signatures must be:

Hard to

- Deny
- Imitate

Easy to

- Generate
- Verify

Digital Signatures: What is the Data I Got?

Just as handwritten signatures, digital signatures must be:

Hard to

- Deny
- Imitate

Easy to

- Generate
- Verify

KeyGen	Given a security parameter k KeyGen outputs a pair $\{pk, sk\}$ of public and secret keys.
Sign	Given a message m and sk, Sign outputs a signature σ.
Verify	Given σ, m, pk, Verify tests if σ is a valid signature of m with respect to pk.

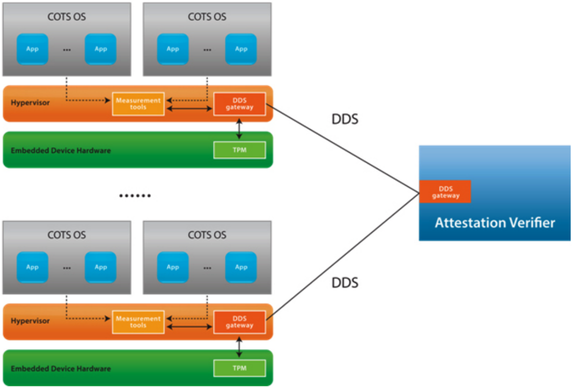
$$\{pk, sk\} \leftarrow \text{KeyGen}(1^k) \qquad \sigma \leftarrow \text{Sign}(sk, m) \qquad \text{Verify}(\sigma, m, pk) = \text{True}$$

Attestation: How Was That Data Computed?

Attestations are mechanisms by which systems (targets) prove their identity to a remote validator.

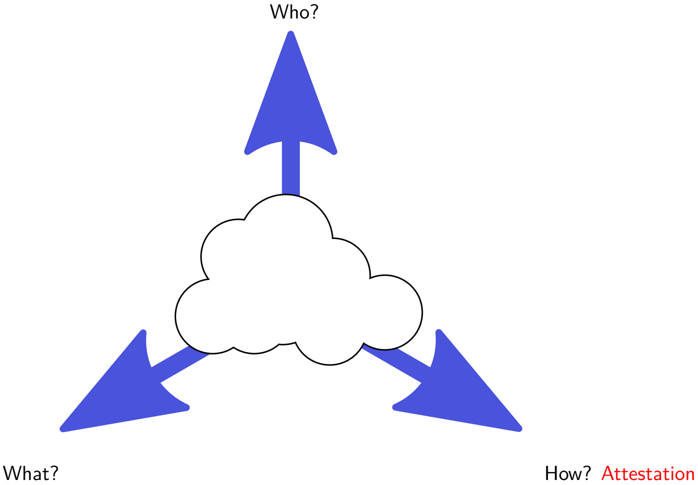
Attestation proves that the target is intact and trustworthy.

Usually achieved by monitoring the target's behavior or the data that it emits.



We now present three of our results.

Non-Interactive Provably Secure Attestations for Arbitrary RSA Prime Generation Algorithms



Context: A Catch-22

The Certification Authority's Catch-22

All today's Certification Authorities face a catch-22:

- Either certify potentially insecure RSA keys or...
- learn the key's factors and hence render these keys insecure.

Analogy: How to test a match without lighting it?

Conversely, Users May Also Distrust Authorities

- Schemes where users need to trust a modulus and use it.

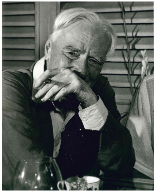
  $\rightarrow$ e.g. Fiat-Shamir.

Conversely, Users May Also Distrust Authorities

- Schemes where users need to trust a modulus and use it.
  $\rightarrow$ e.g. Fiat-Shamir.
- PKI users wanting to check that a root PK was properly generated before using it.

Conversely, Users May Also Distrust Authorities

- Schemes where users need to trust a modulus and use it.
  $\rightarrow$ e.g. Fiat-Shamir.
- PKI users wanting to check that a root PK was properly generated before using it.



"Distrust of authority should be the first civic duty"
Norman Douglas.

In General

How to ascertain, before using or certifying an RSA modulus that this modulus was properly generated?

Previous Proofs of Properties of Composite Moduli

- Van de Graff and Peralta [dGP88]
  $\rightarrow$ n is a Blum integer.
- Boyar, Friedl and Lund [BFL90]
  $\rightarrow$ n is square-free.
- Gennaro, Micciancio and Rabin [GMR98]
  $\rightarrow$ n is a product of quasi-safe primes.
- Camenisch and Michels [CM99]
  $\rightarrow$ n is a product of two safe primes.
- Juels and Guajardo [JJ02]
  $\rightarrow$ n with verifiable randomness.
- Micali [Mic93], Boneh [BF97], Chan [CFT98], Mao [Mao98]
  $\rightarrow$ $n = pq$, without leaking anything but p, q's primality.

Despite All These Results

Fix any arbitrary prime generation algorithm $\mathcal{G}$.

We know no simple (i.e. non theoretical) non-interactive proof that a modulus n contains two prime factors generated by $\mathcal{G}$.

e.g. prove that n has at least two factors of the form:

- $p = x \| \mathrm{SHA}(x)$
- or $p = x \| \mathrm{HELLO\ WORLD} \| y$
- or such that $\lfloor 1/\sin^2(p) \rfloor \bmod 3419 = 17$

In the following slides we will describe such a construction.

Desired Features of the Proof

We wish the modulus attestation scheme to be:

- **Generic:** Work for any prime number generation algorithm $\mathcal{G}$.

Desired Features of the Proof

We wish the modulus attestation scheme to be:

- **Generic:** Work for any prime number generation algorithm $\mathcal{G}$.

- **Secretless:** n is provided with an attestation ω_n that can be verified by everybody without knowing any secret. [similar to signatures]

Desired Features of the Proof

We wish the modulus attestation scheme to be:

- **Generic:** Work for any prime number generation algorithm $\mathcal{G}$.
- **Secretless:** n is provided with an attestation ω_n that can be verified by everybody without knowing any secret. [similar to signatures]
- **Non-interactive:** ω_n, n can be checked without any interaction with the creator of n. [similar to signatures]

Desired Features of the Proof

We wish the modulus attestation scheme to be:

- **Generic:** Work for any prime number generation algorithm $\mathcal{G}$.
- **Secretless:** n is provided with an attestation ω_n that can be verified by everybody without knowing any secret. [similar to signatures]
- **Non-interactive:** ω_n, n can be checked without any interaction with the creator of n. [similar to signatures]
- **Compact:** The size of ω_n should be manageable, i.e. polynomial in $\log n$. [similar to signatures]

Desired Features of the Proof

We wish the modulus attestation scheme to be:

- **Generic:** Work for any prime number generation algorithm $\mathcal{G}$.
- **Secretless:** n is provided with an attestation ω_n that can be verified by everybody without knowing any secret. [similar to signatures]
- **Non-interactive:** ω_n, n can be checked without any interaction with the creator of n. [similar to signatures]
- **Compact:** The size of ω_n should be manageable, i.e. polynomial in $\log n$. [similar to signatures]
- **Efficient:** Calculations for creating or verifying an attestation must remain manageable. [while not "fast", our solution is practical]

Our Construction

1. Generate $k \geq 2$ random numbers $r_1, \ldots, r_k$ and define $h_i = \mathcal{H}(i, r_i)$.

Our Construction

1. Generate $k \geq 2$ random numbers $r_1, \ldots, r_k$ and define $h_i = \mathcal{H}(i, r_i)$.
2. Let $p_i = \mathcal{G}(h_i)$ and $N = \prod_{i=1}^{k} p_i$

Our Construction

1. Generate $k \geq 2$ random numbers $r_1, \ldots, r_k$ and define $h_i = \mathcal{H}(i, r_i)$.
2. Let $p_i = \mathcal{G}(h_i)$ and $N = \prod_{i=1}^{k} p_i$
3. Define $(X_1, X_2) = \mathcal{H}'_2(N)$, where $\mathcal{H}'_2$ is a hash function which outputs two indices $1 \leq X_1 < X_2 \leq k$.

Our Construction

1. Generate $k \geq 2$ random numbers $r_1, \ldots, r_k$ and define $h_i = \mathcal{H}(i, r_i)$.
2. Let $p_i = \mathcal{G}(h_i)$ and $N = \prod_{i=1}^{k} p_i$
3. Define $(X_1, X_2) = \mathcal{H}'_2(N)$, where $\mathcal{H}'_2$ is a hash function which outputs two indices $1 \leq X_1 < X_2 \leq k$.
4. This defines $n = p_{X_1} \times p_{X_2}$ and

$$\omega_n = \{r_1, r_2, \ldots, r_{X_1 - 1}, \star, r_{X_1+1}, \ldots, r_{X_2-1}, \star, r_{X_2+1}, \ldots, r_k\}$$

Our Construction

1. Generate $k \geq 2$ random numbers $r_1, \ldots, r_k$ and define $h_i = \mathcal{H}(i, r_i)$.
2. Let $p_i = \mathcal{G}(h_i)$ and $N = \prod_{i=1}^{k} p_i$
3. Define $(X_1, X_2) = \mathcal{H}'_2(N)$, where $\mathcal{H}'_2$ is a hash function which outputs two indices $1 \leq X_1 < X_2 \leq k$.
4. This defines $n = p_{X_1} \times p_{X_2}$ and

$$\omega_n = \{r_1, r_2, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, r_{X_2-1}, \star, r_{X_2+1}, \ldots, r_k\}$$

Here, a $\star$ denotes a placeholder used to skip one index.

- The data ω_n is called the attestation of n.
- The algorithm $\mathcal{A}$ used to obtain ω_n is called an attestator.

Generating and Validating an Attestation

The choice of the r_i determines N, which is split into two parts $\begin{cases} N/n \\ n \end{cases}$

Security: Splitting is determined by d, which is the digest of N, and is hence unpredictable by the opponent.

$$\left.\begin{array}{ccccc}
r_1 & \xrightarrow{\mathcal{H}(1,r_1)} & h_1 & \xrightarrow{\mathcal{G}(h_1)} & p_1 \\
\vdots & & \vdots & & \vdots \\
r_{X_1} & \xrightarrow{\mathcal{H}(X_1,r_{X_1})} & h_{X_1} & \xrightarrow{\mathcal{G}(h_{X_1})} & p_{X_1} \\
\vdots & & \vdots & & \vdots \\
r_{X_2} & \xrightarrow{\mathcal{H}(X_2,r_{X_2})} & h_{X_2} & \xrightarrow{\mathcal{G}(h_{X_2})} & p_{X_2} \\
\vdots & & \vdots & & \vdots \\
r_k & \xrightarrow{\mathcal{H}(k,r_k)} & h_k & \xrightarrow{\mathcal{G}(h_k)} & p_k
\end{array}\right\} \xrightarrow{\quad\times\quad} N \xrightarrow{\mathcal{H}_2'(N)} d = \{X_1, X_2\}$$

The Attestator $\mathcal{A}$

Input: $r_1, \ldots, r_k$
Output: n, ω_n

$N \leftarrow 1$
for all $i = 1$ to k **do**
  $h_i \leftarrow \mathcal{H}(i, r_i)$
  $p_i \leftarrow \mathcal{G}\left(h_i\right)$
  $N \leftarrow N \times p_i$
end for
$(X_1, X_2) \leftarrow \mathcal{H}'_2(N)$
$\omega_n \leftarrow \{r_1, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, r_{X_2-1}, \star, r_{X_2+1}, \ldots, r_k\}$
$n \leftarrow p_{X_1} \times p_{X_2}$
return n, ω_n

The Validator $\mathcal{V}$

Input: n, ω_n
Output: True or False

The Validator $\mathcal{V}$

Input: n, ω_n
Output: True or False

$N \leftarrow n$
for all $r_i \neq \star$ in ω_n **do**
  $h_i \leftarrow \mathcal{H}(i, r_i)$
  $p_i \leftarrow \mathcal{G}\left(h_i\right)$
  $N \leftarrow N \times p_i$
end for

The Validator $\mathcal{V}$

Input: n, ω_n
Output: True or False

$N \leftarrow n$
for all $r_i \neq \star$ in ω_n **do**
  $h_i \leftarrow \mathcal{H}(i, r_i)$
  $p_i \leftarrow \mathcal{G}(h_i)$
  $N \leftarrow N \times p_i$
end for
$(X_1, X_2) \leftarrow \mathcal{H}'_2(N)$
if $r_{X_1} = \star$ and $r_{X_2} = \star$ and $\#\{r_i \in \omega_n \text{ s.t. } r_i = \star\} = 2$ **then**
  return True
else
  return False
end if

The Issue: Efficiency

- Selecting only two primes out of k makes attacks easy:

  $\rightarrow$ the attacker must only bet on two indexes among k

  $\rightarrow$ i.e. his success probability is $\frac{2}{k(k-1)}$.
- In addition, this is the success probability per trial and attestations are non-interactive experiments, hence with sufficient computing power this can be broken even for large k (e.g. $k = 10^6 \Rightarrow 2^{39}$ security).
- Solutions:
  - Use moduli with more than $\ell = 2$ prime factors.
  - Use more than $u = 1$ modulus for signing or encrypting the same message.

Different (ℓ, u) parameters allow to reach sufficient security.

First Idea: Use More Than Two Factors

Two properly generated p_is suffice to make n factoring-resistant (secure for RSA encryption and signature).

Security grows quickly with the number of factors (details in thesis).

Bigger moduli slows-down RSA cubically.

We hence buy security at the price of slower execution.

Attestator for Moduli Having $\ell \geq 3$ Factors

Attestator $\mathcal{A}$ for moduli having more than two factors ($\ell \geq 3$).

Input: $r_1, \ldots, r_k$

Output: n, ω_n

$N \leftarrow 1$

for all $i \leftarrow 1$ to k **do**

$h_i \leftarrow \mathcal{H}(i, r_i)$

$p_i \leftarrow \mathcal{G}(h_i)$

$N \leftarrow N \times p_i$

end for

$(X_1, \ldots, X_\ell) \leftarrow \mathcal{H}'_\ell(N)$

$\omega_{\mathbf{n}} \leftarrow \{r_1, \ldots, \star \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, \star, \ldots, r_{X_\ell-1}, \star, r_{X_\ell+1}, \ldots, r_k\}$

$n \leftarrow p_{X_1} \times \cdots \times p_{X_\ell}$

return n, ω_n

Second Idea: Use Several Moduli

- Sign u times the same message m using u different moduli.
- **One** properly generated n suffices to get m signed.

Second Idea: Use Several Moduli

Encryption is a bit more tricky.

- Share a secret key $\kappa = \kappa_1 \oplus, \ldots, \oplus \kappa_u$
- Encrypt each share κ_i with a different modulus n_i
- Then encrypt $c = \text{AES}(\kappa, m)$. [Hybrid encryption+Secret sharing]

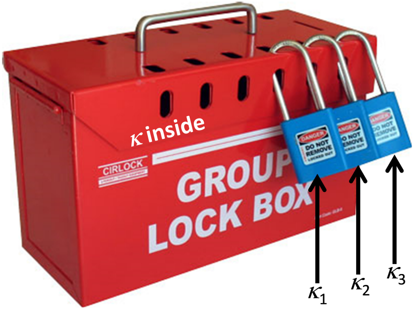
  If at least one κ_i gets encrypted by a properly generated n_i then κ is safe.

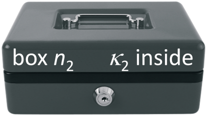
👍 Security grows quickly with the number of moduli (cf. thesis).

👎 Working with u moduli slows-down calculations linearly in u.

We hence buy again security at the price of slower execution.

Second Idea: Encryption Analogy





Second Idea: Encryption Analogy

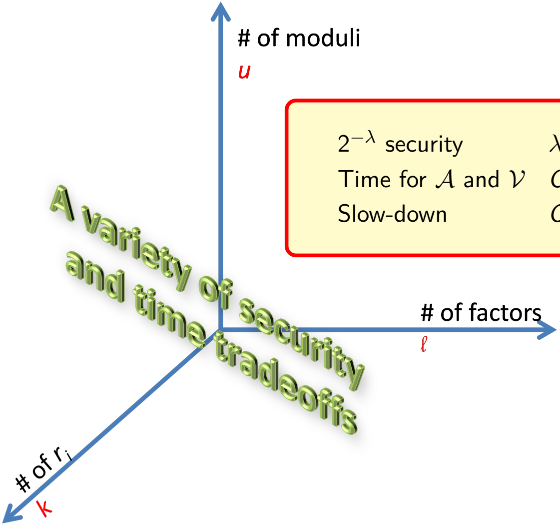
Attestator for Several Moduli ($u \geq 2$)

Attestator $\mathcal{A}$ for $u \geq 2$ bi-factor moduli.

Input: $r_1, \ldots, r_k$
Output: $\mathbf{n} := (n_1, \ldots, n_u), \omega_{\mathbf{n}}$

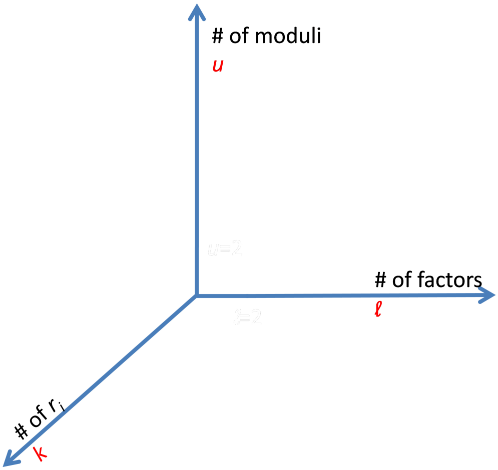
$N \leftarrow 1$
for all $i \leftarrow 1$ to k **do**
  $h_i \leftarrow \mathcal{H}(i, r_i)$
  $p_i \leftarrow \mathcal{G}(h_i)$
  $N \leftarrow N \times p_i$
end for
$(X_1, \ldots, X_{2u}) \leftarrow \mathcal{H}'_{2u}(N)$
$\omega_{\mathbf{n}} \leftarrow \{r_1, \ldots, r_{X_1-1}, \star, r_{X_1+1}, \ldots, \star, \ldots, r_{X_{2u}-1}, \star, r_{X_{2u}+1}, \ldots, \star, \ldots, r_k\}$
for all $j \leftarrow 1$ to u **do**
  $n_j \leftarrow p_{X_{2j}} \times p_{X_{2j+1}}$
end for
return $\mathbf{n} := (n_1, \ldots, n_u), \omega_{\mathbf{n}}$

The General Case: Combine Both Variants ($u > 1, \ell > 2$)

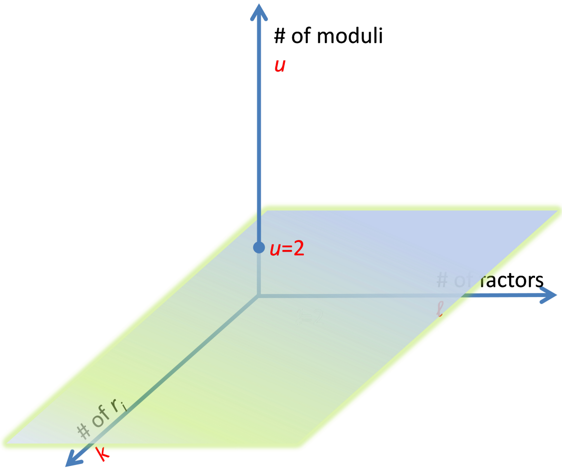


$2^{-\lambda}$ security	$\lambda \simeq (\ell - 1) u \log_2 \left(\frac{\ell u}{k - \ell u}\right)$
Time for $\mathcal{A}$ and $\mathcal{V}$	$O(k \times \text{complexity}(\mathcal{G}))$
Slow-down	$O(\frac{u\ell^3}{8})$

Example for $u = 2$ (2 Moduli, Variable Number of Factors)



Example for $u = 2$ (2 Moduli, Variable Number of Factors)



Example for $u = 2$ (2 Moduli, Variable Number of Factors)

Analogy: 

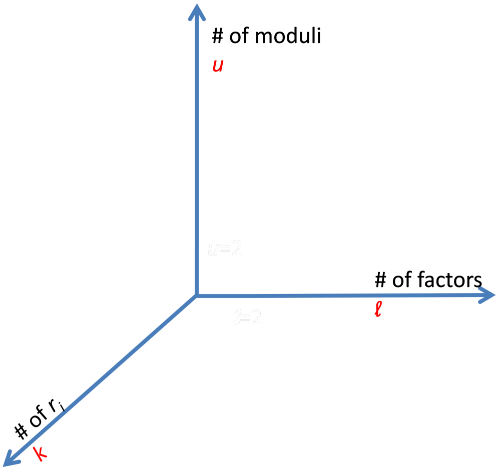
Example: $u = 2$ (two moduli).

Using an attestation of $k = 2^{11}$ elements and $\ell = 10$ factors per modulus we get a security of 2^{-119}.

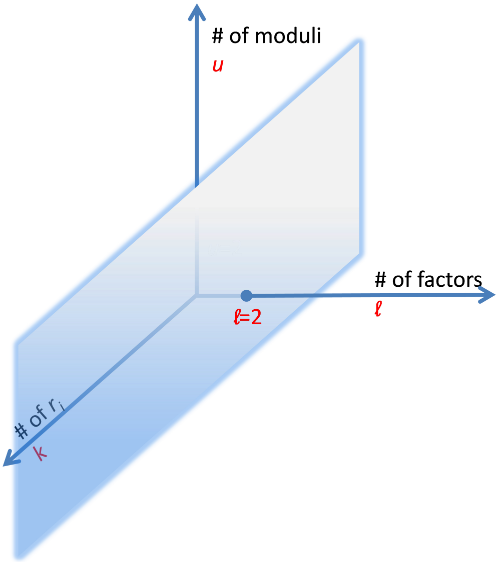
It takes 3.4 minutes to create or validate this attestation on a standard PC.

$\log_2 k$	Time	$\ell = 6$	$\ell = 8$	$\ell = 10$	$\ell = 12$	$\ell = 14$	$\ell = 16$	$\ell = 18$	$\ell = 20$
8	25 s	43	54	64	72	79	84	89	93
9	51 s	53	69	83	95	107	117	126	135
10	1.7 min	64	83	101	118	134	148	162	175
11	3.4 min	74	97	119	140	160	179	197	214
12	6.8 min	84	111	138	162	186	209	231	253
13	13.7 min	94	125	156	185	212	239	266	291
14	27.3 min	104	139	174	207	238	269	300	329
15	54.6 min	114	153	192	229	264	299	334	367
16	1.8 hrs	124	167	210	251	290	329	368	405

Example for $\ell = 2$ (2 Factors, Variable Number of Moduli)



Example for $\ell = 2$ (2 Factors, Variable Number of Moduli)



Example for $\ell = 2$ (2 Factors, Variable Number of Moduli)

Analogy: 

Example: $\ell = 2$ (two factors).

With $k = 2^{12}$ elements and $u = 4$ moduli we get a security of 2^{-195}.

Validated in 6.8 minutes.

$\log_2 k$	Time	$u = 3$	$u = 4$	$u = 5$	$u = 6$
9	51 s	71	109	145	173
10	1.7 min	87	138	193	246
11	3.4 min	102	167	239	315
12	6.8 min	117	195	285	383
13	13.7 min	132	223	330	450
14	27.3 min	147	251	375	516
15	54.6 min	162	279	420	582
16	1.8 hrs	177	307	465	648

Compressing the Attestation

$$
\begin{tikzcd}[column sep=small, row sep=tiny]
 & & & r_{111} = h(r_{11},1) \\
 & & r_{11} = h(r_1,1) \arrow[ur] \arrow[dr] & \\
 & & & r_{110} = h(r_{11},0) \\
 & r_1 = h(r,1) \arrow[uur] \arrow[ddr] & & \\
 & & & r_{101} = h(r_{10},1) \\
 & & r_{10} = h(r_1,0) \arrow[ur] \arrow[dr] & \\
 & & & r_{100} = h(r_{10},0) \\
r \arrow[uuuur] \arrow[ddddr] & & & \\
 & & & r_{011} = h(r_{01},1) \\
 & & r_{01} = h(r_0,1) \arrow[ur] \arrow[dr] & \\
 & & & r_{010} = h(r_{01},0) \\
 & r_0 = h(r,0) \arrow[uur] \arrow[ddr] & & \\
 & & & r_{001} = h(r_{00},1) \\
 & & r_{00} = h(r_0,0) \arrow[ur] \arrow[dr] & \\
 & & & r_{000} = h(r_{00},0)
\end{tikzcd}
$$

Compressing the Attestation

Red: secret nodes.

r

$r_1 = h(r, 1)$

$r_{11} = h(r_1, 1)$

$r_{111} = h(r_{11}, 1)$

$r_{110} = h(r_{11}, 0)$

$r_{10} = h(r_1, 0)$

$r_{101} = h(r_{10}, 1)$

$r_{100} = h(r_{10}, 0)$

$r_0 = h(r, 0)$

$r_{01} = h(r_0, 1)$

$r_{011} = h(r_{01}, 1)$

$r_{010} = h(r_{01}, 0)$

$r_{00} = h(r_0, 0)$

$r_{001} = h(r_{00}, 1)$

$r_{000} = h(r_{00}, 0)$

Compressing the Attestation

Red: secret nodes. Blue: revealed nodes.

r

$r_1 = h(r, 1)$

$r_0 = h(r, 0)$

$r_{11} = h(r_1, 1)$

$r_{10} = h(r_1, 0)$

$r_{01} = h(r_0, 1)$

$r_{00} = h(r_0, 0)$

$r_{111} = h(r_{11}, 1)$

$r_{110} = h(r_{11}, 0)$

$r_{101} = h(r_{10}, 1)$

$r_{100} = h(r_{10}, 0)$

$r_{011} = h(r_{01}, 1)$

$r_{010} = h(r_{01}, 0)$

$r_{001} = h(r_{00}, 1)$

$r_{000} = h(r_{00}, 0)$

Compressing the Attestation

Red: secret nodes. Blue: revealed nodes.

r

$r_1 = h(r, 1)$

$r_0 = h(r, 0)$

$r_{11} = h(r_1, 1)$

$r_{10} = h(r_1, 0)$

$r_{01} = h(r_0, 1)$

$r_{00} = h(r_0, 0)$

$r_{111} = h(r_{11}, 1)$

$r_{110} = h(r_{11}, 0)$

$r_{101} = h(r_{10}, 1)$

$r_{100} = h(r_{10}, 0)$

$r_{011} = h(r_{01}, 1)$

$r_{010} = h(r_{01}, 0)$

$r_{001} = h(r_{00}, 1)$

$r_{000} = h(r_{00}, 0)$

Compressing the Attestation

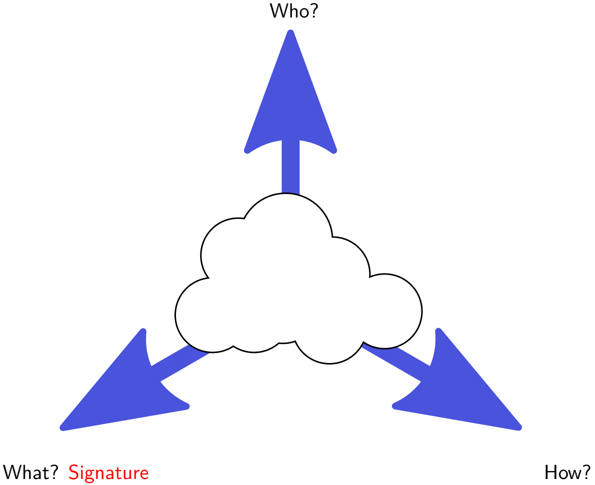
Red: secret nodes. Blue: revealed nodes.

$$
\begin{tikzcd}[column sep=small, row sep=tiny]
 & & & r_{111} = h(r_{11},1) \\
 & & r_{11} = h(r_1,1) \arrow[ur] \arrow[dr] & \\
 & & & r_{110} = h(r_{11},0) \\
 & r_1 = h(r,1) \arrow[uur] \arrow[ddr] & & \\
 & & & r_{101} = h(r_{10},1) \\
 & & r_{10} = h(r_1,0) \arrow[ur] \arrow[dr] & \\
 & & & r_{100} = h(r_{10},0) \\
 r \arrow[uuuur] \arrow[ddddr] & & & \\
 & & & r_{011} = h(r_{01},1) \\
 & & r_{01} = h(r_0,1) \arrow[ur] \arrow[dr] & \\
 & & & r_{010} = h(r_{01},0) \\
 & r_0 = h(r,0) \arrow[uur] \arrow[ddr] & & \\
 & & & r_{001} = h(r_{00},1) \\
 & & r_{00} = h(r_0,0) \arrow[ur] \arrow[dr] & \\
 & & & r_{000} = h(r_{00},0)
\end{tikzcd}
$$

Finally: $\mathcal{A}$ and $\mathcal{V}$ Are Parallel-Friendly

z processors $\Rightarrow$ $\times z$ speedup

Legally Fair Contract Signing



Desirable Properties

The two following properties are desirable in contract signing protocols:

Viability

If both parties follow the protocol properly, then at its termination each party will have his counterpart's signature on the contract.

Desirable Properties

The two following properties are desirable in contract signing protocols:

Viability

If both parties follow the protocol properly, then at its termination each party will have his counterpart's signature on the contract.

Fairness

If one party, say Alice, follows the protocol properly then Bob has Alice's signature on the contract iff Alice also has Bob's signature on the contract.

Fairness: Gradual Release vs. Trusted Third Party

Lots of prior work on fairness.

In essence two big ideas:

- Fairness via Trusted Third Party (TTP):
  - Fast protocol execution
  - Online TTP is impractical

    $\rightarrow$ communication bottleneck
  - If participants are honest $\Rightarrow$ offline TTP
- Gradual Release:
  - No need for a TTP
  - Assumes equal computational power for participants
  - Long protocol execution even if participants are honest

Our Work and New Results

We introduce a novel form of fairness without TTPs called *legal fairness* defined as follows:

Legal Fairness

Any transferable proof of involvement tying one party to a message, also ties the other party to the message.

Our idea

Verifiers will be given the means to determine when Alice tries to involve Bob.

When this happens, verifiers will contact Bob who will be able to prove Alice's involvement.

Schnorr Signatures

The proposed signature paradigm is based on Schnorr signatures.

g is a generator of $\mathbb{G}$ such as $\mathbb{G}$ is cyclic group of prime order q.

Let m the message to be signed.

Signer		**Verifier**
Has secret-key x		Uses the public-key $y \leftarrow g^x$
Pick $k \in_{\mathrm{r}} \mathbb{Z}_q$ $r \leftarrow g^k$ $e \leftarrow H(m, r)$ $s \leftarrow k - ex \bmod q$		
	$\xrightarrow{\quad r,s \quad}$	Check the signature (s, e) : $r == g^s y^e$ and $H(m, r) == e$

Classical Schnorr Signatures and the Forking Lemma

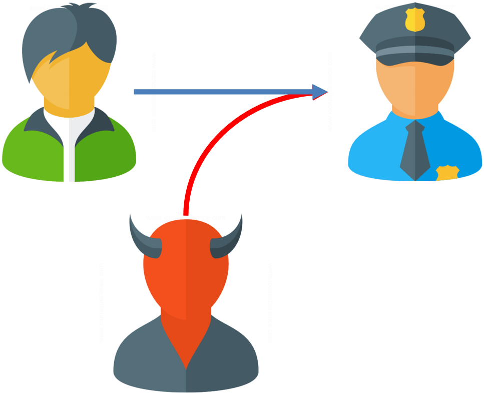
- Pointcheval and Stern [PS96]: DLP + ROM $\Rightarrow$ Schnorr is secure
- Pointcheval and Stern establish that in the ROM, the opponent can obtain from the forger two valid forgeries $\{\ell, s, e\}$ and $\{\ell, s', e'\}$ for the same oracle query $\{m, r\}$ but with different digests $e \neq e'$.

  Hence, $r = g^s y^{-e} = g^{s'} y^{-e'}$ allows to compute the DL of $y = g^x$.

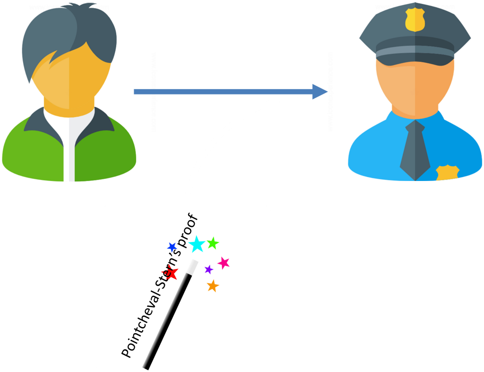
  Indeed:

$$g^s y^{-e} = g^{s'} y^{-e'} \;\Rightarrow\; y = g^{\frac{s'-s}{e-e'}} \;\Rightarrow\; \mathrm{Dlog}_g(y) = \frac{s'-s}{e-e'}$$

The Threat



Pointcheval-Stern's Proof

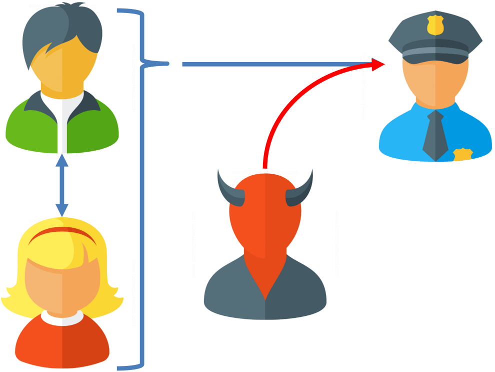


Schnorr Co-Signatures

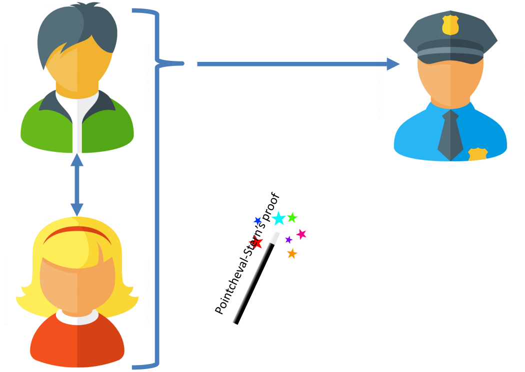
Alice		Bob
Read Bob's directory entry		Read Alice's directory entry
$y_{A,B} \leftarrow y_A \times y_B, k_A \in_R \mathbb{Z}_q^*$		$y_{A,B} \leftarrow y_A \times y_B, k_B \in_R \mathbb{Z}_q^*$
$r_A \leftarrow g^{k_A}$		$r_B \leftarrow g^{k_B}$
	$\xleftarrow{\quad\rho\quad}$	$\rho \leftarrow H(0\|r_B)$
	$\xrightarrow{\quad r_A\quad}$	
if $H(0\|r_B) \neq \rho$ **then** abort	$\xleftarrow{\quad r_B\quad}$	
$r \leftarrow r_A \times r_B$		$r \leftarrow r_A \times r_B$
$e \leftarrow H(1\|m\|r)$		$e \leftarrow H(1\|m\|r)$
$s_A \leftarrow k_A - ex_A \bmod q$		$s_B \leftarrow k_B - ex_B \bmod q$
if s_B is incorrect **then** abort	$\xleftarrow{\quad s_B\quad}$	
$s \leftarrow s_A + s_B \bmod q$	$\xrightarrow{\quad s_A\quad}$	$s \leftarrow s_A + s_B \bmod q$

r, s is verified by checking that: $r == g^s y_{A,B}^e$ and $H(m, r) == e$

Pointcheval-Stern's Proof Extends to Co-Signatures



Pointcheval-Stern's Proof Extends to Co-Signatures



Schnorr Co-Signatures: The Fairness Problem!

Alice		**Bob**
Read Bob's directory entry		Read Alice's directory entry
$y_{A,B} \leftarrow y_A \times y_B, k_A \in_R \mathbb{Z}_q^*$		$y_{A,B} \leftarrow y_A \times y_B, k_B \in_R \mathbb{Z}_q^*$
$r_A \leftarrow g^{k_A}$		$r_B \leftarrow g^{k_B}$
	$\xleftarrow{\rho}$	$\rho \leftarrow H(0\|r_B)$
	$\xrightarrow{r_A}$	
if $H(0\|r_B) \neq \rho$ **then** abort	$\xleftarrow{r_B}$	
$r \leftarrow r_A \times r_B$		$r \leftarrow r_A \times r_B$
$e \leftarrow H(1\|m\|r)$		$e \leftarrow H(1\|m\|r)$
$s_A \leftarrow k_A - ex_A \bmod q$		$s_B \leftarrow k_B - ex_B \bmod q$
if s_B is incorrect **then** abort	$\xleftarrow{s_B}$	
$s \leftarrow s_A + s_B \bmod q$	$\xrightarrow{s_A}$	$s \leftarrow s_A + s_B \bmod q$

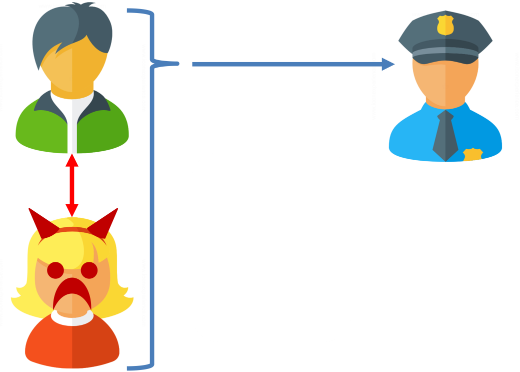
r, s is verified by checking that: $r == g^s y_{A,B}^e$ and $H(m, r) == e$

Schnorr Co-Signatures: The Fairness Problem!

Alice		Bob
Read Bob's directory entry		Read Alice's directory entry
$y_{A,B} \leftarrow y_A \times y_B, k_A \in_R \mathbb{Z}_q^*$		$y_{A,B} \leftarrow y_A \times y_B, k_B \in_R \mathbb{Z}_q^*$
$r_A \leftarrow g^{k_A}$		$r_B \leftarrow g^{k_B}$
	$\xleftarrow{\rho}$	$\rho \leftarrow H(0\|r_B)$
	$\xrightarrow{r_A}$	
if $H(0\|r_B) \neq \rho$ **then** abort	$\xleftarrow{r_B}$	
$r \leftarrow r_A \times r_B$		$r \leftarrow r_A \times r_B$
$e \leftarrow H(1\|m\|r)$		$e \leftarrow H(1\|m\|r)$
$s_A \leftarrow k_A - ex_A \bmod q$		$s_B \leftarrow k_B - ex_B \bmod q$
if s_B is incorrect **then** abort	$\xleftarrow{s_B}$	
$s \leftarrow s_A + s_B \bmod q$	$\xrightarrow{s_A}$	$s \leftarrow s_A + s_B \bmod q$
		if s_A is incorrect **then** too bad!

r, s is verified by checking that: $r == g^s y_{A,B}^e$ and $H(m, r) == e$

Here Alice Attacks Bob



Legally Fair Contract Signing

A new type of threat $\Rightarrow$ requires a new solution!

Our solution: the concept of *legally fair contract signing*.
We assume that:

- Bob is stateful. i.e. Bob keeps in an internal nonvolatile memory $\mathcal{L}$ traces of *problematic* sessions.
- Alice uses a second digital signature algorithm σ.

Schnorr Co-Signatures

Alice		**Bob**
	$\xleftrightarrow{\text{share } m, y_{A,B}}$	
$k_A \in_R \mathbb{Z}_q^*, r_A \leftarrow g^{k_A}$		$k_B \in_R \mathbb{Z}_q^*, r_B \leftarrow g^{k_B}$
	$\xleftarrow{\rho}$	$\rho \leftarrow H(0\\|r_B)$
	$\xrightarrow{r_A}$	
if $H(0\\|r_B) \neq \rho$ **then** abort	$\xleftarrow{r_B}$	
$r \leftarrow r_A \times r_B$		$r \leftarrow r_A \times r_B$
$e \leftarrow H(1\\|m\\|r)$		$e \leftarrow H(1\\|m\\|r)$
$s_A \leftarrow k_A - ex_A \bmod q$		$s_B \leftarrow k_B - ex_B \bmod q$
if s_B is incorrect **then** abort	$\xleftarrow{s_B}$	
	$\xrightarrow{s_A}$	**if** s_A is incorrect **then** abort
$s \leftarrow s_A + s_B \bmod q$		$s \leftarrow s_A + s_B \bmod q$

Legally Fair Contract Signing

Alice		Bob
	share $m, y_{A,B}$ $\longleftrightarrow$	
$k_A \in_R \mathbb{Z}_q^*, r_A \leftarrow g^{k_A}$		$k_B \in_R \mathbb{Z}_q^*, r_B \leftarrow g^{k_B}$
	$\xleftarrow{\rho}$	$\rho \leftarrow H(0\Vert r_B)$
$t \leftarrow \sigma(r_A\Vert\text{Alice}\Vert\text{Bob})$	$\xrightarrow{r_A, t}$	**if** t is incorrect **then** abort
if $H(0\Vert r_B) \neq \rho$ **then** abort	$\xleftarrow{r_B}$	
$r \leftarrow r_A \times r_B$ $e \leftarrow H(1\Vert m\Vert r\Vert\text{Alice}\Vert\text{Bob})$ $s_A \leftarrow k_A - ex_A \bmod q$		$r \leftarrow r_A \times r_B$ $e \leftarrow H(1\Vert m\Vert r\Vert\text{Alice}\Vert\text{Bob})$ $s_B \leftarrow k_B - ex_B \bmod q$ store t, s_B in $\mathcal{L}$
if s_B is incorrect **then** abort	breakpoint 1 $\xleftarrow{s_B}$	
	breakpoint 2 $\xrightarrow{s_A}$	**if** s_A is incorrect **then** abort
$s \leftarrow s_A + s_B \bmod q$		$s \leftarrow s_A + s_B \bmod q$ **if** $\{m, r, s\}$ is valid erase $\mathcal{L}$

Intuition of the Legal Fairness Proof

We present here the intuition, the formal proof is in the thesis.

- *Before* breakpoint 1: Nothing bad can possibly happen $\rightarrow$ Because no information depending on m was released by any of the parties.

Intuition of the Legal Fairness Proof

We present here the intuition, the formal proof is in the thesis.

- *Before* breakpoint 1: Nothing bad can possibly happen $\rightarrow$ Because no information depending on m was released by any of the parties.

- *After* breakpoint 1: Bob can misbehave (go silent or send a bad s_B) $\rightarrow$ In such a case Alice will detect this and punish him (she will just shut-up).

Intuition of the Legal Fairness Proof

We present here the intuition, the formal proof is in the thesis.

- *Before* breakpoint 1: Nothing bad can possibly happen $\rightarrow$ Because no information depending on m was released by any of the parties.

- *After* breakpoint 1: Bob can misbehave (go silent or send a bad s_B) $\rightarrow$ In such a case Alice will detect this and punish him (she will just shut-up).

- *At the* breakpoint 2: Is critical. If Bob did not misbehave we hit the core issue: $\rightarrow$ Here Alice has the final say.

  She can hence stop sending information or send wrong information.

  We need to show that if this happens Bob can either:

  - **Case A** deny involvement or **case B** involve Alice as well.
  - Note that outcomes depend on the way in which Alice tries to use the information she has in her possession.

Case (A): Denial by Bob

Case (A): Alice exhibits r and s_B:

- Bob will pretend that:
  1. Alice picked s_B, r at random
  2. Alice computed $e \leftarrow H(1\|m\|r\|\text{Alice}\|\text{Bob})$
  3. Alice computed $r_B \leftarrow g^{s_B} y_B^e$
  4. Alice computed $r_A \leftarrow r \times r_B^{-1}$
  5. Alice signed $t \leftarrow \sigma(r_A\|\text{Alice}\|\text{Bob})$.
- Indeed $g^{s_B} y_B^e = r_B$ and this looks exactly as if Bob produced s_B, r_B.
- Note that this would *not* be the case if Alice manages to prove that r_A is not random, *i.e.* that she knows the DL of r_A. $\rightarrow$ This is case (B).

Case (B): Alice Shoots a Bullet Into Her Own Leg

Case (B): Alice proves that she knows the DL of r_A:

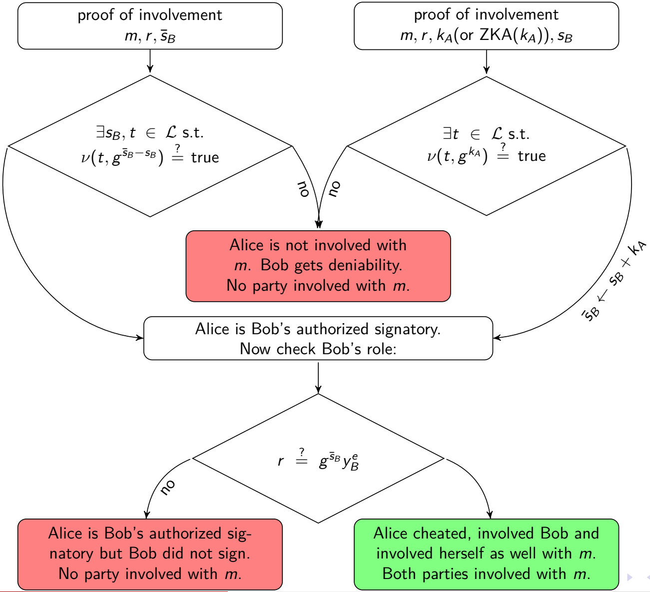
- A verifier seeing that is instructed to contact Bob and ask him for t.
- The verifier now has a signature:
  - t of r_A
  - a proof that Alice knows the DL of r_A.

We *legally define* this state of knowledge as **a deliberate agreement** on Alice's behalf to any message signed by Bob, and in particular m.

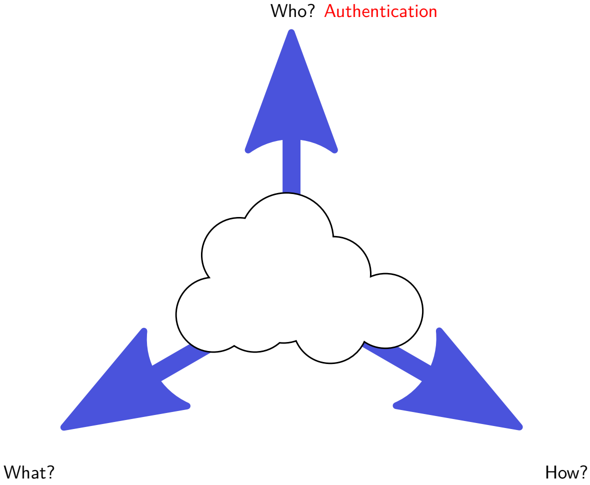
Legal Fairness = "Legal" Definition:

If Alice ever publishes $\sigma(g^k \| \text{Alice} \| \text{Bob})$ and k then this means **by definition** that Alice declares her deliberate agreement to any message signed by Bob.

Schematically



Thrifty ZK: Linear Programming & Cryptography



Recall Σ-Protocols

We use the mathematical framework of Σ-protocols:

$$\mathcal{P} \quad \begin{array}{c} \xrightarrow{\quad x \quad} \\ \xleftarrow{\quad c \quad} \\ \xrightarrow{\quad y \quad} \end{array} \quad \mathcal{V}$$

Σ-protocols have a very important property: If a cheater correctly bets on the value of the challenge c he can get accepted without knowing the secret-key.

Security vs. Work in Σ-Protocols

Consider a Σ-protocol.

Security Level

The *security level* S is defined as the challenge min-entropy

$$S := -\log_2 \max_c \Pr(c)$$

Informally, S is the security level (log of successful cheating probability) corresponding to the attacker's most favorable challenge.

Work Factor

The *work factor* W is defined as the expected (average) value of the prover's working time $W(x, c)$:

$$W := \mathbb{E}_{x,c}\left[W(x, c)\right]$$

Informally, W is $\mathcal{P}$'s average work during the protocol.

Obtained Security Per Work Done $\Rightarrow$ Security Efficiency

Two Traditional Concerns: Increase S. Decrease W.

The Actual Problem: Increase E!

Security Efficiency

The *security efficiency* E, is defined as the ratio between S and W:

$$E := \frac{S}{W}$$

In other words, E is the amount of security bits per operation provided by the protocol at its current parameter setting.

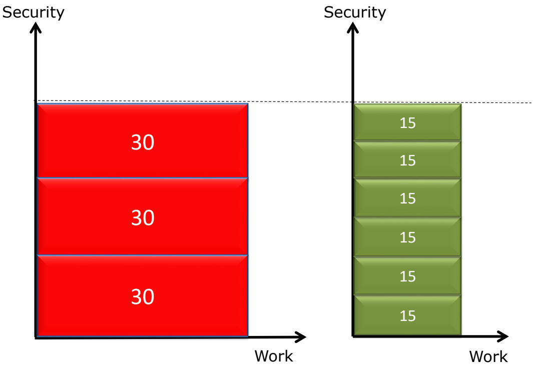
What Can We Control? The Challenge Probabilities

The idea:

Because the answer to different challenges requires different amounts of calculation, we can favor the challenges for which answers are fast.

This will decrease both S and W but might increase $E = S/W$.
Finding the best E is a linear programming problem.

Graphic Illustration



In the above diagrams each rectangle represents a protocol round.

In both cases the same overall security level (90 bits) is achieved.

Either by 3 heavy steps of 30 bits or by 6 lighter steps of 15 bits (thrifty version).

$$\text{Surface}(\text{Red}) > \text{Surface}(\text{Green})$$

Fiat-Shamir

The relationship between the secret-keys s_i and the public-keys v_i is:

$$s_i^2 v_i = 1 \bmod n$$

Prover $\mathcal{P}$		**Verifier** $\mathcal{V}$
Has k secrets s_i		Uses k public-keys v_i
Pick $r \in_R \mathbb{Z}_n$ $x \leftarrow r^2$		
	$\xrightarrow{x}$	
		Pick $c \in_R \{0,1\}^k$
	$\xleftarrow{c}$	
$y \leftarrow r \prod_{i=0}^{k-1} s_i^{c_i}$		
	$\xrightarrow{y}$	
		Check if $y^2 \prod_{i=0}^{k-1} v_i^{c_i} \stackrel{?}{=} x$

An Example: Fiat-Shamir for $k = 3$

In Fiat-Shamir response to a challenge c costs a number of multiplications equal to c's Hamming weight.

Take 3-bit challenges as an example.

Trivially, the number of k-bit challenges having Hamming weight j is $\binom{k}{j}$

$$\underbrace{000}_{\text{1 value}} \quad \underbrace{001 \;\; 010 \;\; 100}_{\text{3 values}} \quad \underbrace{011 \;\; 101 \;\; 110}_{\text{3 values}} \quad \underbrace{111}_{\text{1 value}}$$

Red: multiplications. Blue: # of c values.

Example: Fiat-Shamir for $k = 3$

Green: HammingWeight(c) = Red: multiplications. Blue: # of c values.

$$W = p_0 \times 0 \times 1 + p_1 \times 1 \times 3 + p_2 \times 2 \times 3 + p_3 \times 3 \times 1 = 3p_1 + 6p_2 + 3p_3$$

Because $p_0 = p_1 = p_2 = p_3 = \frac{1}{8}$

$$W = 3 \times \frac{1}{8} + 6 \times \frac{1}{8} + 3 \times \frac{1}{8} = \frac{12}{8} = 1.5$$

Therefore the corresponding efficiency is $E = \frac{3}{1.5} = 2$ bits per multiplication.

Example: Thrifty Fiat-Shamir for $k = 3$

Green: HammingWeight(c) = Red: multiplications. Blue: # of c values.

$$W = p_0 \times 0 \times 1 + p_1 \times 1 \times 3 + p_2 \times 2 \times 3 + p_3 \times 3 \times 1 = 3p_1 + 6p_2 + 3p_3$$

We degrade security by giving the attacker the possibility to bet on a challenge whose probability is $\epsilon > 1/8$.

$$\text{Given } \epsilon, \begin{cases} \text{minimize} & W = 3p_1 + 6p_2 + 3p_3 \\ \text{subject to} & 0 \leq p_0, p_1, p_2, p_3 \leq \epsilon \\ & p_0 + 3p_1 + 3p_2 + p_3 = 1 \end{cases}$$

Let $p_0 = p_1 = p_2 = \epsilon$, and $p_3 = 1 - 7\epsilon$, yielding a work factor of

$$W = 9\epsilon + 3(1 - 7\epsilon) = 3(1 - 4\epsilon)$$

Therefore the corresponding efficiency is $E = \frac{-\log_2 \epsilon}{3(1-4\epsilon)}$, which at $\epsilon = 1/7$ equals $7 \log_2 7/9 \simeq 2.18$.

10% improvement over standard Fiat-Shamir.

Other Scientific Results.

Other Thesis Publications

- Slow Motion Zero Knowledge Identifying with Colliding Commitment
  H. Ferradi, R. Géraud, D. Naccache. [Inscrypt'15]
- Public-Key Based Lightweight Swarm Authentication
  S. Cogliani, B. Feng, H. Ferradi, R. Géraud, D. Maimuţ, D. Naccache, R. Portella do Canto, G. Wang. [Cryptology ePrint Archive]
- Compact CCA2-secure Hierarchical Identity-Based Broadcast Encryption for Fuzzy-entity Data Sharing
  Weiran, L. Jianwei, W. Qianhong, B. Qin, D. Naccache, H. Ferradi. [J. Infor. Sci.]
- When Organized Crime Applies Academic Results
  H. Ferradi, R. Géraud, D. Naccache, A. Tria. [J. of Cryptographic Engineering]
- Human Public-Key Encryption
  H. Ferradi, R. Géraud, D. Naccache. [Mycrypt'16]
- Honey Encryption for Language: Robbing Shannon to Pay Turing?
  M. Beunardeau, H. Ferradi, R. Géraud, D. Naccache. [Mycrypt'16]

Security Publications (Not in the Thesis)

- Secure Application Execution in Mobile Devices
  M. Msgna, H. Ferradi, R. Akram, K. Markantonakis. [New Codebreakers'16]

- Backtracking-Assisted Multiplication
  H. Ferradi, R. Géraud, D. Maimuţ, D. Naccache, H. Zhou. [ArcticCrypt'16]

- Communicating Covertly through CPU Monitoring
  J.-M. Cioranesco, H. Ferradi, D. Naccache. [IEEE S&P'13]

- Process Table Covert Channels: Exploitation and Countermeasures
  J.-M. Cioranesco, H. Ferradi, R. Géraud, D. Naccache. [Cryptology ePrint Archive]

Thank you for your attention!

References I

Dan Boneh and M. Franklin.

Efficient generation of shared RSA keys.

In *Advances in Cryptology – CRYPTO'97*, pages 425–439. Springer Verlag, 1997.

J. Boyar, K. Friedl, and C. Lund.

Practical zero-knowledge proofs: Giving hints and using deficiencies.

In *Advances in Cryptology – EUROCRYPT'89*, pages 155–172. Springer Berlin Heidelberg, 1990.

A. Chan, Y. Frankel, and Y. Tsiounis.

Easy come - easy go divisible cash.

In *Advances in Cryptology – EUROCRYPT'98*, pages 561–575. Springer-Verlag, 1998.

References II

J. Camenisch and M. Michels.
Proving that a number is the product of two safe primes.
In *Advances in Cryptology – EUROCRYPT'99*, pages 107–122. Springer-Verlag, 1999.

J. Van de Graaf and R. Peralta.
A simple and secure way to show the validity of your public key.
In *Advances in Cryptology – CRYPTO'87*, pages 128–134. Springer Berlin Heidelberg, 1988.

S. Goldwasser, S. Micali, and C. Rackoff.
The Knowledge Complexity of Interactive Proof-Systems.
In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC'85, pages 291–304. ACM, 1985.

References III

R. Gennaro, D. Micciancio, and T. Rabin.

An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products.

In *Proceedings of the 5th ACM conference on Computer and Communications Security*, pages 67–72. ACM, 1998.

A. Juels and J.Guajardo.

RSA key generation with verifiable randomness.

In *Public Key Cryptography*, pages 357–374. Springer Berlin Heidelberg, 2002.

W. Mao.

Verifiable partial sharing of integer factors.

In *Selected Areas in Cryptography – SAC'98*, pages 94–105. Springer-Verlag, 1998.

References IV

S. Micali.

Fair public key cryptosystems.

In *Advances in Cryptology – CRYPTO'92*, pages 113–138. Springer Berlin Heidelberg, 1993.

David Pointcheval and Jacques Stern.

Security Proofs for Signature Schemes, pages 387–398.

Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.