Shaken, not Stirred: Automated Discovery of Subtle Attacks on Protocols using Mix-Nets

Jannik Dreier Pascal Lafourcade Dherkra Mahmoud

August 2024

Designing Secure Schemes is Difficult!

How can we be **convinced** that a **protocol** is a **good** one?

Designing Secure Schemes is Difficult!

How can we be **convinced** that a **protocol** is a **good** one?

Publish the protocol and wait until someone finds an attack.

Designing Secure Schemes is Difficult!

How can we be **convinced** that a **protocol** is a **good** one?

Publish the protocol and wait until someone finds an attack.

Prove that there is no attack.

Designing Secure Schemes is Difficult!

How can we be **convinced** that a **protocol** is a **good** one?

Publish the protocol and wait until someone finds an attack.

Prove that there is no attack.

Usual problems with proofs:

- proving is a difficult task,
- pencil-and-paper proofs are error-prone.

How can we be **convinced** that a **proof** is a **good** one?

Designing Secure Schemes is Difficult!

How can we be **convinced** that a **protocol** is a **good** one?


Publish the protocol and wait until someone finds an attack.


Prove that there is no attack.

Usual problems with proofs:

- proving is a difficult task,
- pencil-and-paper proofs are error-prone.

How can we be **convinced** that a **proof** is a **good** one?


Publish the proof and wait until someone finds a mistake.

Designing Secure Schemes is Difficult!

How can we be **convinced** that a **protocol** is a **good** one?


Publish the protocol and wait until someone finds an attack.


Prove that there is no attack.

Usual problems with proofs:

- proving is a difficult task,
- pencil-and-paper proofs are error-prone.

How can we be **convinced** that a **proof** is a **good** one?


Publish the proof and wait until someone finds a mistake.


Computer-Aided Security: ProVerif

Shaken, not Stirred

Automated Discovery of Subtle Attacks on Protocols using Mix-Nets

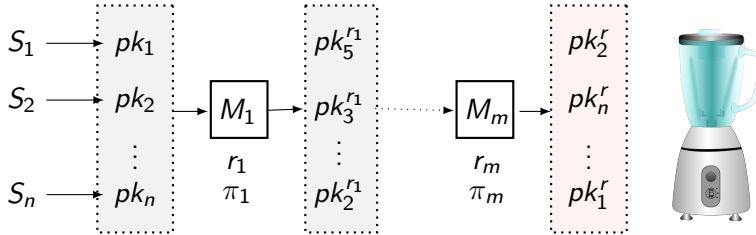
Shaken

Stirred

Exponentiation Mix-Nets

Haenni *et al.* USENIX'11

El Gamal, $pk_i = g^{sk_i}$



$$\pi = \prod_{i=1}^{m} \pi_i,\ r = \prod_{i=1}^{m} r_i$$

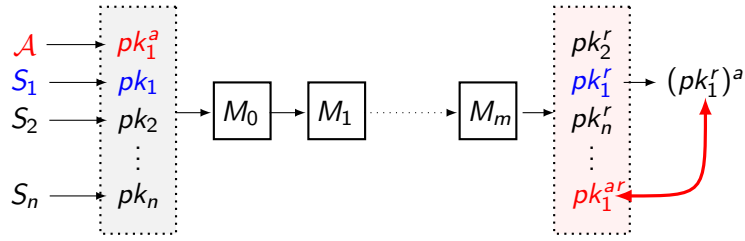
with sk_i, everyone can check $(g^r)^{sk_i} = pk_i^r$

For anonymity and unlinkability of voters

Shaken

Stirred

Attack Exponentiation Mix-Nets: Pfitzmann 1994, Rakeei *et al.* 2022

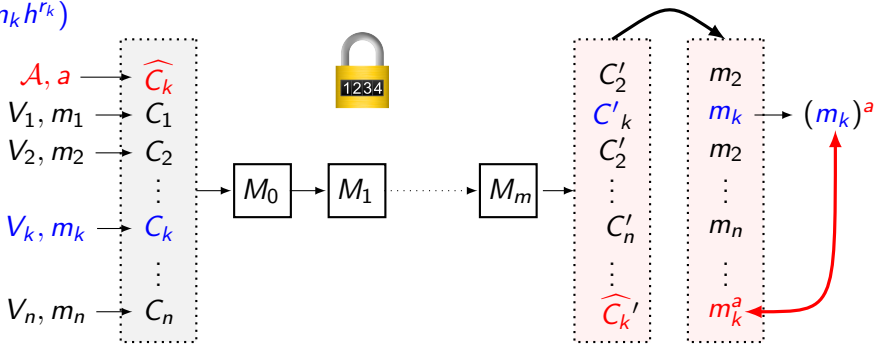


Attack against Re-encryption Mix-Nets Park et al. 1994 for voting

Candidates are public

$C_k = (g^{r_k}, m_k h^{r_k})$

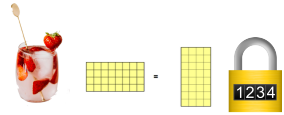
Decryption by the vote authority



$$\widehat{C_k} = (C_k)^a = (g^{r_k a}, (m_k h^{r_k})^a)) = (g^{r_k a}, m_k{}^a h^{r_k a}))$$

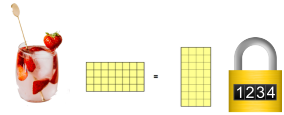
Contributions

Can we find **automatically** such “cryptographic” attacks?



Contributions

Can we find **automatically** such "cryptographic" attacks?



ProVerif models for Mixnet:

- Exponentiation
- ElGamal
- Weak and Strong NIZKP

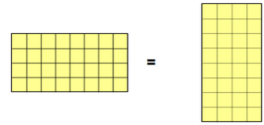
Applications:

e-voting

e-exam

Crypto Santa

Exponentiation and Signature Modeling



Exponentiation

$(g^x)^y = (g^y)^x$

$$\begin{aligned}((g^x)^y)^z &= ((g^x)^z)^y \\ &= ((g^z)^x)^y \\ &= ((g^z)^y)^x \\ &= ((g^y)^z)^x \\ &= ((g^y)^x)^z\end{aligned}$$

$$exp(exp(g,x),y) = exp(exp(g,y),x)$$

$$exp(exp(exp(g,x),y),z) = exp(exp(exp(g,x),z),y)$$

Signature

$pk = g^{sk}$, $\sigma = sign(m, g, sk)$, $checksign(\sigma, pk) = m$

$$getmess(sign(m, X, sk)) = m$$

$$checksign(sign(m, X, sk), X, exp(X, sk)) = m$$

ElGamal Encryption Modeling

ElGamal

- Encryption and decryption

$pk = g^{sk},\ c = (g^r, (g^{sk})^r m)$ $\qquad dec(enc(m, X, exp(X, sk), r), X, sk) = m$

- Re-Encryption with $g^{r'}$
  $c' = (g^{r'} g^r, g^{r'} g^{sk} m)$

$$reenc(enc(m, X, exp(X, sk), r), r', X, exp(X, sk)) = enc(m, X, exp(X, sk), sum(r, r'))$$

$c^a = ((g^r)^a, (g^{sk})^a m^a)$

$$EXP(enc(m, X, exp(X, x), r), a) = enc(exp(m, a), X, exp(X, x), mult(r, a))$$

Non-Interactive ZKP (NIZKP)

Weak NIZKP

Public parameter: $pk = g^{sk}$

- Construction with sk and a random
  $A = g^a$, $c = H(A)$, $f = a + c.sk$, $\pi = (c, f)$
- Verification of $\pi = (c, f)$ with pk, check $H(g^f \cdot pk^{-c}) \stackrel{?}{=} c$
  $H(g^f \cdot pk^{-c}) = H(g^{a+c \cdot sk} \cdot g^{-sk \cdot c}) = H(g^a) = c$

Fake a Weak NIZKP

- Construction with A' and f' two randoms
  $c' = H(A')$ and produce $\pi' = (c', f')$
- Verification for $pk' = (g^{f'} \cdot A'^{-1})^{c'^{-1}}$
  $H(g^{f'} \cdot pk'^{-c'}) = H(g^{f'} \cdot ((g^{f'} \cdot A'^{-1})^{c'^{-1}})^{-c'}) = H(A') = c'$

Allows attack against Exponentiation Mix-Nets with Weak NIZKP

Strong NIZKP: $c = H(A, pk)$

Non-Interactive ZKP

NIZPK attack: Link of pk with $pk' = pk^{-c'^{-1}}$
$A' = g^{f'}.pk$ and $c' = H(A')$ then $\pi' = (c', f')$

Verification of $\pi' = (c', f')$ with pk', check $H(g^{f'}.pk'^{-c'}) \stackrel{?}{=} c'$:
$H(g^{f'}.pk'^{-c'}) = H(A'.pk^{-1}(pk^{-c'^{-1}})^{-c'}) = H(A'.pk^{-1}.pk) = H(A')$

ProVerif Modelling

- Weak ZKP: $c = H(A)$

$$check(wzkp(A, X, sk), X, exp(X, sk), H(A)) = true$$

- Strong ZKP: $c = H(A, pk)$

$$check(szkp(A, g, sk), g, exp(g, sk), H(g, exp(g, sk), A)) = true$$

Weak modeling allows the intruder to choose the value of the public key !

Results on Mixnets

Protocol	ZKP	Result	Time
Exponentiation Mix-Nets	without	✗	2 s
	weak	✗	1 m 6 s
	strong	✓	3 s
Re-encryption Mix-Nets	without	✗	1 s
	weak	✗	2 s
	strong	✓	1 s

Applications

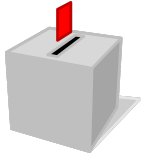
Protocol	ZKP	Property	Result	Time
Remark! 	without	Anonymous Marking	✗	3 m 16 s
		Anonymous Examiner	✗	4 m 19 s
	weak	Anonymous Marking	✗	9 m 35 s
		Anonymous Examiner	✗	9 m 23 s
	strong	Anonymous Marking	✓	11 s
		Anonymous Examiner	✓	7 s
Haenni Voting 	without	Vote Privacy	✗	4 m 35 s
	weak		✗	9 m 35 s
	strong		✓	14 s
	weak	Anonymous Shuffling	✗	4 m 6 s
	strong		✓	9 s
Estonian IVXV 	without	Vote Privacy	✗	1 s
	weak		✗	25 s
	strong		✓	8 s

Conclusion

New ProVerif models for:

- Exponentiation Mixnets
- Re-Encryption Mixnets
- Weak ZKP
- Strong ZKP
- ElGamal
- Signature

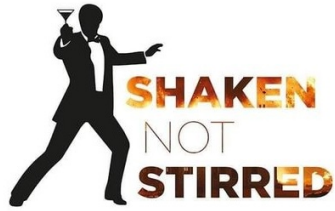
Applications



e-voting

e-exam

Crypto Santa



Thanks for your attention!