Batch Zero-Knowledge Proofs

Rolf Haenni

http://e-voting.bfh.ch

Seminar, E-Voting Group, BFH

November 21st, 2012

Outline

Outline

Motivation

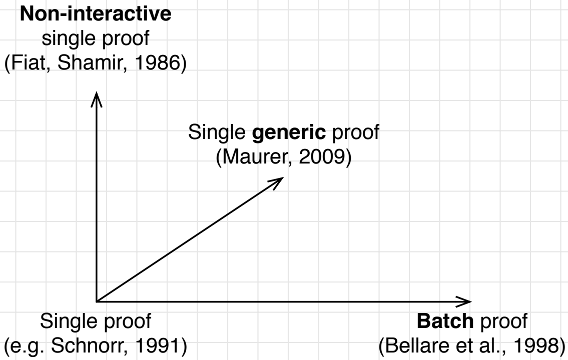
Proving multiple instances of the same type of zero-knowledge proof can be done in two ways

1. Standard AND-composition proof
   - → Linear proof size (commitment, response)
   - → Linear running time (generation, verification)
2. Batch Proof
   - → Constant proof size
   - → Improved running time

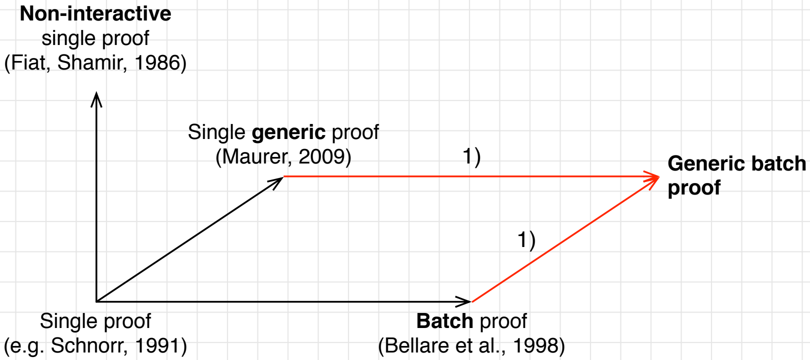
Applications

- Multi-decryption of ciphertexts $c_1, \ldots, c_n$
  - → $c_i = (a_i, b_i) =$ ElGamal ciphertext
  - → $m_i = b_i \cdot a_i^{-x} =$ decryption with private key x
- Multi-committment to messages $m_1, \ldots, m_n$ (e.g. commiting to a matrix)
  - → $m_i = i$-th column vector of M
  - → $c_i = C(m_i, s_i) =$ extended Pedersen commitment of m_i
- Multi-blinding values $x_1, \ldots, x_n$
  - → $z =$ common blinding value
  - → $x_i' = x_i^z =$ blinding of x_i
- Commitment multiplication proof of length > 2

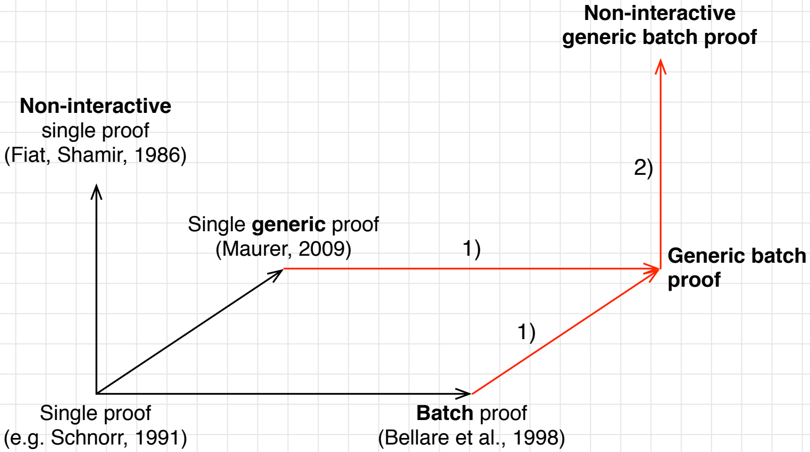
Overview


Non-interactive
single proof
(Fiat, Shamir, 1986)
Single **generic** proof
(Maurer, 2009)
Single proof
(e.g. Schnorr, 1991)
Batch proof
(Bellare et al., 1998)

Overview


Non-interactive
single proof
(Fiat, Shamir, 1986)
Single **generic** proof
(Maurer, 2009)
1)
Generic batch proof
1)
Single proof
(e.g. Schnorr, 1991)
Batch proof
(Bellare et al., 1998)

Overview



Non-interactive
generic batch proof
Non-interactive
single proof
(Fiat, Shamir, 1986)
2)
Single **generic** proof
(Maurer, 2009)
1)
Generic batch
proof
1)
Single proof
(e.g. Schnorr, 1991)
Batch proof
(Bellare et al., 1998)

References

M. Bellare, J. A. Garay, and T. Rabin.
Batch verification with applications to cryptography and checking.
LATIN'98: 3rd Latin American Symposium on Theoretical Informatics, Campinas, Brazil, 1998.

R. Aditya, K. Peng, C. Boyd, E. Dawson, and B. Lee.
Batch verification for equality of discrete logarithms and threshold decryptions.
ACNS'04, 2nd International Conference on Applied Cryptography and Network Security, Yellow Mountain, China, 2004.

K. Peng, C. Boyd, and E. Dawson.
Batch zero-knowledge proof and verification and its applications.
ACM Transactions on Information and System Security, 10(2), 2007.

Related Work

U. Maurer.
Unifying zero-knowledge proofs of knowledge.
AFRICACRYPT'09, 2nd International Conference on Cryptology in Africa, Gammarth, Tunisia, 2009.

D. Wikström.
A Commitment-Consistent Proof of a Shuffle.
ACISP'09, 14th Australasian Conference on Information Security and Privacy, Brisbane, Australia, 2009.

B. Terelius and D. Wikström.
Proofs of Restricted Shuffles.
AFRICACRYPT'10, 3rd International Conference on Cryptology in Africa, Stellenbosch, South Africa, 2010.

Outline

Non-Interactive Preimage Proof

- Let $(X, +, 0)$ and $(Y, \times, 1)$ be groups of finite order
- Consider a one-way group homomorphism $\phi : X \to Y$
- Let $b = \phi(a)$ be publicly known
- The prover P proves knowledge of a using the Σ-protocol:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $t = \phi(\omega)$
  3. Compute $c = H(b, t)$
  4. Compute $s = \omega + c \cdot a$
  5. Publish $\pi = (t, s)$
- To verify π, the verifier V computes $c = H(b, t)$ and checks $\phi(s) \stackrel{?}{=} t \cdot b^c$

Outline

Example 1: Discrete Logarithm (Schnorr)

- Let g be a generator of G_q
- Let $c = g^m$ be a publicly known commitment of $m \in \mathbb{Z}_q$
- P proves knowledge of m using the Σ-protocol for:

$$a = m,$$
$$b = c,$$
$$\phi(x) = g^x,$$

where $\phi : \underbrace{\mathbb{Z}_q}_{X} \to \underbrace{G_q}_{Y}$

Example 2: Pedersen Commitment

- Let g and h be generators of G_q
- Let $c = g^m h^s$ be a publicly known Pedersen commitment of $m \in \mathbb{Z}_q$ with randomization $s \in \mathbb{Z}_q$
- P proves knowledge of m and s using the Σ-protocol for:

$$a = (m, s),$$
$$b = c,$$
$$\phi(x_1, x_2) = g^{x_1} h^{x_2},$$

where $\phi : \underbrace{\mathbb{Z}_q \times \mathbb{Z}_q}_{X} \to \underbrace{G_q}_{Y}$

- Note that $\omega = (\omega_1, \omega_2)$ and $s = (s_1, s_2)$, but t is a single value

Example 3: Equality of Discrete Logarithms

- Let g_1 and g_2 be generators of G_q
- Let $c_1 = g_1^m$ and $c_2 = g_2^m$ be public commitments of $m \in \mathbb{Z}_q$
- P proves knowledge of m using the Σ-protocol for:

$$a = m,$$
$$b = (c_1, c_2),$$
$$\phi(x) = (g_1^x, g_2^x),$$

where $\phi : \underbrace{\mathbb{Z}_q}_{X} \to \underbrace{G_q \times G_q}_{Y}$

- Note that $t = (t_1, t_2)$, but ω and s are single values

Outline

Composed Proofs

- Consider n one-way group homomorphisms $\phi_i : X_i \to Y_i$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a_i)$
- P proves knowledge of $a_1, \ldots, a_n$ using the Σ-protocol for:

$$
\begin{aligned}
a &= (a_1, \ldots, a_n), \\
b &= (b_1, \ldots, b_n), \\
\phi(x_1, \ldots, x_n) &= (\phi_1(x_1), \ldots, \phi_n(x_n)),
\end{aligned}
$$

where $\phi : \underbrace{X_1 \times \cdots \times X_n}_{X} \to \underbrace{Y_1 \times \cdots \times Y_n}_{Y}$

- Note that $\omega = (\omega_1, \ldots, \omega_n)$, $t = (t_1, \ldots, t_n)$, $s = (s_1, \ldots, s_n)$

Composed Proofs: Performance

- Proof $\pi = (t, s)$ has size $O(n)$
- Generation: $O(n)$
  - → Let $\bar{r}$ be the average number of modExps in ϕ_i
  - → Computing $\phi(\omega)$ requires $n \cdot \bar{r}$ modExps
  - → $modExps(n) = n \cdot \bar{r}$
- Verification: $O(n)$
  - → Computing $\phi(s)$ requires $n \cdot \bar{r}$ modExps
  - → Computing b^c requires n modExps
  - → $modExps(n) = n \cdot (\bar{r} + 1)$
- Remark: c is usually small (e.g., 160 bits for SHA-1)

Special Case 1: Common Function

- Consider a single one-way group homomorphism $\phi : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi(a_i)$
- P proves knowledge of $a_1, \ldots, a_n$ using the Σ-protocol for:

$$a = (a_1, \ldots, a_n),$$
$$b = (b_1, \ldots, b_n),$$
$$\phi(x_1, \ldots, x_n) = (\phi(x_1), \ldots, \phi(x_n)),$$

where $\phi : \underbrace{X \times \cdots \times X}_{X} \to \underbrace{Y \times \cdots \times Y}_{Y}$

- Note that $\omega = (\omega_1, \ldots, \omega_n)$, $t = (t_1, \ldots, t_n)$, $s = (s_1, \ldots, s_n)$
- Proof size and performance remain unchanged

Special Case 2: Common Preimage

- Consider n one-way group homomorphisms $\phi_i : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a using the Σ-protocol for:

$$\begin{aligned} & a \\ & b = (b_1, \ldots, b_n), \\ & \phi(x) = (\phi_1(x), \ldots, \phi_n(x)), \end{aligned}$$

where $\phi : X \to \underbrace{Y \times \cdots \times Y}_{Y}$

- Note that $t = (t_1, \ldots, t_n)$, but ω and s are single values
- Improved (linear) proof size, performance remains unchanged

Outline

Outline

Type-1 Batch Proof

- Consider a single one-way group homomorphism $\phi : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi(a_i)$
- P proves knowledge of $a_1, \ldots, a_n$ as follows:
  - → V chooses $e_1, \ldots, e_n$ uniformly at random

$$b = \prod_i b_i^{e_i} = \prod_i \phi(a_i)^{e_i} = \prod_i \phi(e_i a_i) = \phi(\sum_i e_i a_i)$$

  - → P generates a preimage proof $\pi = (t, s)$ for $a = \sum_i e_i a_i$ and $b = \phi(a)$
- V computes $b = \prod_i b_i^{e_i}$ and verifies π
- Note that P does not need to compute $b = \prod_i b_i^{e_i}$

Type-1 Batch Proof: Non-Interactive Version

- Consider a single one-way group homomorphism $\phi : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi(a_i)$
- P proves knowledge of $a_1, \ldots, a_n$ as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $t = \phi(\omega)$
  3. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n$
  4. Compute $a = \sum_i e_i a_i$
  5. Compute $c = H(b, t)$
  6. Compute $s = \omega + c \cdot a$
  7. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $c = H(b_1, \ldots, b_n, t)$, $b = \prod_i b_i^{e_i}$, and checks $\phi(s) \stackrel{?}{=} t \cdot b^c$

Type-1 Batch Proof: Non-Interactive Version

- Consider a single one-way group homomorphism $\phi : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi(a_i)$
- P proves knowledge of $a_1, \ldots, a_n$ as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $t = \phi(\omega)$
  3. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n$
  4. Compute $a = \sum_i e_i a_i$
  5. Compute $c = H(b, t)$ $\Leftarrow$ we don't want P to compute b
  6. Compute $s = \omega + c \cdot a$
  7. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $c = H(b_1, \ldots, b_n, t)$, $b = \prod_i b_i^{e_i}$, and checks $\phi(s) \stackrel{?}{=} t \cdot b^c$

Type-1 Batch Proof: Non-Interactive Version

- Consider a single one-way group homomorphism $\phi : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi(a_i)$
- P proves knowledge of $a_1, \ldots, a_n$ as follows:

   1. Choose $\omega \in_R X$ uniformly at random
   2. Compute $t = \phi(\omega)$
   3. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n$
   4. Compute $a = \sum_i e_i a_i$
   5. Compute $c = H(b_1, \ldots, b_n, t)$
   6. Compute $s = \omega + c \cdot a$
   7. Publish $\pi = (t, s)$

- V computes $e_i = H(b_i, t)$, $c = H(b_1, \ldots, b_n, t)$, $b = \prod_i b_i^{e_i}$, and checks $\phi(s) \stackrel{?}{=} t \cdot b^c$

Type-1 Batch Proof: Performance

- Proof $\pi = (t, s)$ has size $O(1)$
- Generation: $O(1)$
  - → Let r be the number of modExps in ϕ
  - → Assume that $a = \sum_i e_i a_i$ can be computed efficiently
  - → $modExps(n) = r$
- Verification: $O(n)$
  - → Computing $b = \prod_i b_i^{e_i}$ requires n modExps
  - → Computing $\phi(s)$ requires r modExps
  - → Computing b^c requires 1 modExp
  - → $modExps(n) = n + r + 1$
- Remark: $e_1, \ldots, e_n$ and c are usually small (e.g., 160 bits)

Outline

Type-2 Batch Proof

- Consider n one-way group homomorphisms $\phi_i : X \rightarrow Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:
  - → V chooses $e_1, \ldots, e_n$ uniformly at random

$$b = \prod_i b_i^{e_i} = \prod_i \phi_i(a)^{e_i} = \prod_i \phi_i(e_i \cdot a)$$

  - → P generates a preimage proof $\pi = (t, s)$ for $b = \phi_e(a)$, where $\phi_e(x) = \prod_i \phi_i(e_i \cdot x)$ is a new one-way group homomorphism
- V computes $b = \prod_i b_i^{e_i}$ and verifies π
- Again, P does not need to compute $b = \prod_i b_i^{e_i}$

Type-2 Batch Proof: Non-Interactive Version

- Consider n one-way group homomorphisms $\phi_i : X \rightarrow Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $t = \phi_e(\omega)$
  3. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n$
  4. Compute $c = H(b_1, \ldots, b_n, t)$
  5. Compute $s = \omega + c \cdot a$
  6. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $b = \prod_i b_i^{e_i}$, $c = H(b, t)$, and checks $\phi_e(s) \stackrel{?}{=} t \cdot b^c$

Type-2 Batch Proof: Non-Interactive Version

- Consider n one-way group homomorphisms $\phi_i : X \to Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $t = \phi_e(\omega)$ ⇐ ϕ_e depends on e_i
  3. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n$
  4. Compute $c = H(b_1, \ldots, b_n, t)$
  5. Compute $s = \omega + c \cdot a$
  6. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $b = \prod_i b_i^{e_i}$, $c = H(b, t)$, and checks $\phi_e(s) \stackrel{?}{=} t \cdot b^c$

Type-2 Batch Proof: Non-Interactive Version

- Consider n one-way group homomorphisms $\phi_i : X \rightarrow Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n$
  3. Compute $t = \phi_e(\omega)$
  4. Compute $c = H(b_1, \ldots, b_n, t)$
  5. Compute $s = \omega + c \cdot a$
  6. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $b = \prod_i b_i^{e_i}$, $c = H(b, t)$, and checks $\phi_e(s) \stackrel{?}{=} t \cdot b^c$

Type-2 Batch Proof: Non-Interactive Version

- Consider n one-way group homomorphisms $\phi_i : X \rightarrow Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:

  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $e_i = H(b_i, t)$ for $i = 1, \ldots, n \Leftarrow e_i$ depend on t
  3. Compute $t = \phi_e(\omega)$
  4. Compute $c = H(b_1, \ldots, b_n, t)$
  5. Compute $s = \omega + c \cdot a$
  6. Publish $\pi = (t, s)$

- V computes $e_i = H(b_i, t)$, $b = \prod_i b_i^{e_i}$, $c = H(b, t)$, and checks $\phi_e(s) \stackrel{?}{=} t \cdot b^c$

Type-2 Batch Proof: Non-Interactive Version

- Consider n one-way group homomorphisms $\phi_i : X \rightarrow Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $e_i = H(b_i)$ for $i = 1, \ldots, n$
  3. Compute $t = \phi_e(\omega)$
  4. Compute $c = H(b_1, \ldots, b_n, t)$
  5. Compute $s = \omega + c \cdot a$
  6. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $b = \prod_i b_i^{e_i}$, $c = H(b, t)$, and checks $\phi_e(s) \stackrel{?}{=} t \cdot b^c$

Type-2 Batch Proof: Non-Interactive Version

- Consider n one-way group homomorphisms $\phi_i : X \rightarrow Y$
- Let $b_1, \ldots, b_n$ be publicly known, where $b_i = \phi_i(a)$
- P proves knowledge of a as follows:
  1. Choose $\omega \in_R X$ uniformly at random
  2. Compute $e_i = H(b_i)$ for $i = 1, \ldots, n$ $\Leftarrow$ Is this secure?
  3. Compute $t = \phi_e(\omega)$
  4. Compute $c = H(b_1, \ldots, b_n, t)$
  5. Compute $s = \omega + c \cdot a$
  6. Publish $\pi = (t, s)$
- V computes $e_i = H(b_i, t)$, $b = \prod_i b_i^{e_i}$, $c = H(b, t)$, and checks $\phi_e(s) \stackrel{?}{=} t \cdot b^c$

Type-2 Batch Proof: Performance

- Proof $\pi = (t, s)$ has size $O(1)$
- Generation: $O(n)$
  - → Let $\bar{r}$ be average the number of modExps in ϕ_i
  - → Computing $\phi_e(\omega)$ requires $n \cdot \bar{r}$ modExps
  - → $modExps(n) = n \cdot \bar{r}$
- Verification: $O(n)$
  - → Computing $b = \prod_i b_i^{e_i}$ requires n modExps
  - → Computing $\phi_e(s)$ requires $n \cdot \bar{r}$ modExps
  - → Computing b^c requires 1 modExp
  - → $modExps(n) = n \cdot (\bar{r} + 1) + 1$
- Remark: $e_1, \ldots, e_n$ and c are usually small (e.g., 160 bits)

Recapitulation

		modExps(n)			
		small exponents	regular exponents	total	
Composition	Generation	–	$n \cdot \bar{r}$	$n \cdot \bar{r}$	$O(n)$
	Verification	n	$n \cdot \bar{r}$	$n \cdot (\bar{r} + 1)$	$O(n)$
Type-1	Generation	–	r	r	$O(1)$
	Verification	$n + 1$	r	$n + r + 1$	$O(n)$
Type-2	Generation	–	$n \cdot \bar{r}$	$n \cdot \bar{r}$	$O(n)$
	Verification	$n + 1$	$n \cdot \bar{r}$	$n \cdot (\bar{r} + 1) + 1$	$O(n)$

Outline

Square and Multiply Algorithm (SMA)

- General idea for computing x^e efficiently (in a semigroup)

$$x^e = \begin{cases} 1, & \text{if } e = 0 \\ (x^{e/2})^2, & \text{if } e \text{ is even} \\ x \cdot (x^{(e-1)/2})^2, & \text{if } e \text{ is odd} \end{cases}$$

$$= \begin{cases} 1, & \text{if } e = 0 \\ x^{e \bmod 2} \cdot (x^{\lfloor e/2 \rfloor})^2, & \text{otherwise} \end{cases}$$

- Let $L = \log e$ denote the bit length of e
- SMA uses L multiplications and L squarings (worst case)
- Total multiplications: $2L$

Square and Multiply for Products of Powers

- To compute a product of powers $\prod_{i=1}^{n} x_i^{e_i}$, SMA uses $n{\cdot}L$ multiplications and $n{\cdot}L$ squarings
- $n-1$ multiplications are needed for the final result
- Total multiplications: $2{\cdot}n{\cdot}L + n - 1 = n \cdot (2L+1) - 1$
- SMA for products of power (in a commutative semigroup)

$$\prod_i x_i^{e_i} = \begin{cases} 1, & \text{if } e_1 = \cdots = e_n = 0 \\ \prod_i \left(x_i^{e_i \bmod 2} \cdot (x_i^{\lfloor e_i/2 \rfloor})^2 \right), & \text{otherwise} \end{cases}$$

Square and Multiply for Products of Powers

- To compute a product of powers $\prod_{i=1}^{n} x_i^{e_i}$, SMA uses $n{\cdot}L$ multiplications and $n{\cdot}L$ squarings
- $n-1$ multiplications are needed for the final result
- Total multiplications: $2{\cdot}n{\cdot}L + n - 1 = n \cdot (2L+1) - 1$
- SMA for products of power (in a commutative semigroup)

$$\prod_i x_i^{e_i} = \begin{cases} 1, & \text{if } e_1 = \cdots = e_n = 0 \\ \prod_i x_i^{e_i \bmod 2} \cdot \prod_i (x_i^{\lfloor e_i/2 \rfloor})^2, & \text{otherwise} \end{cases}$$

Square and Multiply for Products of Powers

- To compute a product of powers $\prod_{i=1}^{n} x_i^{e_i}$, SMA uses $n \cdot L$ multiplications and $n \cdot L$ squarings
- $n - 1$ multiplications are needed for the final result
- Total multiplications: $2 \cdot n \cdot L + n - 1 = n \cdot (2L + 1) - 1$
- SMA for products of power (in a commutative semigroup)

$$\prod_i x_i^{e_i} = \begin{cases} 1, & \text{if } e_1 = \cdots = e_n = 0 \\ \prod_i x_i^{e_i \bmod 2} \cdot \left(\prod_i x_i^{\lfloor e_i/2 \rfloor}\right)^2, & \text{otherwise} \end{cases}$$

- Uses $n \cdot L$ multiplications and L squarings
- Total multiplications: $(n + 1) \cdot L$

Outline

Prime Order Co-Domain

- For a batch proof to be sound, ϕ (resp. ϕ_e) must have a prime order co-domain Y
- Otherwise, if $|Y|$ is composite, Y may contain low-order sub-groups
- For example, let $G_2 = \{x_1, x_2\} \subseteq Y$ be a sub-group of Y
- If we pick two integers $e_1, e_2 \in \mathbb{Z}$ at random, then

$$P(x_1^{e_1} = x_2^{e_2}) = \frac{1}{2}$$

- Therefore, the probability that an incorrect input $b_i \in G_2$ can pass the verification is $\frac{1}{2}$

Prime Order Sub-Group of $\mathbb{Z}_p^*$

- To avoid this problem, let the co-domain of ϕ (resp. ϕ_e) be a prime order sub-group $G_q \subset \mathbb{Z}_p^*$, where $p = k \cdot q + 1$
- If we pick two integers $e_1, e_2 \in \mathbb{Z}$ at random, then

$$P(x_1^{e_1} = x_2^{e_2}) = \frac{1}{q}$$

  for any distinct values $x_1, x_2 \in G_q$
- Therefore, batch proofs are sound for high-order sub-groups $G_q \subset \mathbb{Z}_p^*$

Testing for Group Membership

- Great, working with prime order co-domains $G_q \subset \mathbb{Z}_p^*$ seems to work. But what if $b_i \notin G_q$?
- For example, let $p = 2 \cdot q + 1$ be a safe prime. Then $x \in G_q$ implies
  - → $(p - x) \in \mathbb{Z}_n^* \setminus G_q$
  - → $x^2 = (p - x)^2$
  - → $x^e = (p - x)^e$ for $e \geq 2$
- Therefore, V needs to test group membership $b_i \in G_q$ for all public inputs $b_i = \phi(a_i)$ resp. $b_i = \phi_i(a)$
- Testing group membership in $G_q \subset \mathbb{Z}_p^*$ requires one modExp

Recapitulation: Update

		$modExps(n)$			
		small exponents	regular exponents	total	
Composition	Generation	–	$n \cdot \bar{r}$	$n \cdot \bar{r}$	$O(n)$
	Verification	n	$n \cdot \bar{r}$	$n \cdot (\bar{r} + 1)$	$O(n)$
Type-1	Generation	–	r	r	$O(1)$
	Verification	$n + 1$	$n + r$	$2n + r + 1$	$O(n)$
Type-2	Generation	–	$n \cdot \bar{r}$	$n \cdot \bar{r}$	$O(n)$
	Verification	$n + 1$	$n \cdot (\bar{r} + 1)$	$n \cdot (\bar{r} + 2) + 1$	$O(n)$

Outline

Conclusion

- There are two types of generic batch proofs
- Type-1 Proof: Common Function
  - → Proof size: $O(1)$
  - → Proof generation runs in $O(1)$ time
  - → Verification not significantly improved
  - → Examples: multi-commitment, multi-encryption, etc.
- Type-2 Proof: Common Preimage
  - → Proof size: $O(1)$
  - → Proof generation and verification not significantly improved
  - → Examples: multi-decryption, multi-blinding, etc.
- Caution: proofs work only for prime order co-domains

Open Quesions

- Missing security proof for the generic proof construction
- Dependency problem in the non-interactive Type-2 Proof
- Missing security proof for the non-interactive version (Fiat-Shamir, random oracle model)
- Implementation in UniCrypt
- Useful for UniVote?