Algorithms of Information Security: Cryptographic Protocols II

Faculty of Information Technology
Czech Technical University in Prague

November 2, 2022

Guillou-Quisquater (GQ) identification protocol

- The Guillou-Quisquater (GQ) identification scheme is an extension of the Fiat-Shamir protocol.
- It allows a reduction in both the number of messages exchanged and memory requirements for user secrets and, like Fiat-Shamir, is suitable for applications in which the claimant has limited power and memory.
- It involves three messages between a claimant A whose identity is to be corroborated, and a verifier B.

Guillou-Quisquater (GQ) identification protocol

Algorithm 1 Guillou-Quisquater (GQ) identification protocol

SUMMARY. A proves its identity (via knowledge of s_A) to B in a 3-pass protocol.

1. *Selection of system parameters.*
    - An authority T, trusted by all parties with respect to binding identities to public keys, selects secret random RSA-like primes p and q yielding a modulus $n = pq$. (As for RSA, it must be computationally infeasible to factor n.)
    - T defines a public exponent $v \geq 3$ with $\gcd(v, \phi) = 1$ where $\phi = (p-1)(q-1)$, and computes its private exponent $s = v^{-1}$ mod ϕ.
    - System parameters (v, n) are made available (with guaranteed authenticity) for all users.

Guillou-Quisquater (GQ) identification protocol

Algorithm 3 Guillou-Quisquater (GQ) identification protocol

---

2. *Selection of per-user parameters.*
   - Each entity A is given a unique identity I_A, from which (the redundant identity) $J_A = f(I_A)$, satisfying $1 < J_A < n$, is derived using a known redundancy function f.
   - T gives to A the secret (accreditation data) $s_A = (J_A)^{-s} \bmod n$.
3. *Protocol messages.* Each of t rounds has three messages as follows (often $t = 1$).
   - $A \to B : I_A,\ x = r^v \bmod n$
   - $A \leftarrow B : e$ (where $1 \leq e \leq v$)
   - $A \to B : y = r \cdot {s_A}^e \bmod n$

Guillou-Quisquater (GQ) identification protocol

Algorithm 3 Guillou-Quisquater (GQ) identification protocol

4. *Protocol actions.* A proves its identity to B by t executions of the following; B accepts the identity only if all t executions are successful.
    - A selects a random secret integer r (the commitment), $1 \leq r \leq n-1$, and computes (the witness) $x = r^v \bmod n$.
    - A sends to B the pair of integers (I_A, x).
    - B selects and sends to A a random integer e (the challenge), $1 \leq e \leq v$.
    - A computes and sends to B (the response) $y = r \cdot {s_A}^e \bmod n$.
    - B receives y, constructs J_A from I_A using f (see above), computes $z = {J_A}^e \cdot y^v \bmod n$, and accepts A's proof of identity if both $z = x$ and $z \neq 0$. (The latter precludes an adversary succeeding by choosing $r = 0$.)

Schnorr identification protocol

- Schnorr identification protocol is an alternative to the Fiat-Shamir and GQ protocols. Its security is based on the intractability of the discrete logarithm problem.
- The design allows pre-computation, reducing the real-time computation for the claimant to one multiplication modulo a prime q; it is thus particularly suitable for claimants of limited computational ability.
- A further important computational efficiency results from the use of a subgroup of order q of the multiplicative group of integers modulo p, where $q \mid (p-1)$; this also reduces the required number of transmitted bits.

Schnorr identification protocol

- Finally, the protocol was designed to require only three passes, and a low communications bandwidth (e.g., compared to Fiat-Shamir).
- The basic idea is that A proves knowledge of a secret a (without revealing it) in a time-variant manner (depending on a challenge e), identifying A through the association of a with the public key v via A's authenticated certificate.

Schnorr identification protocol

Algorithm 4 Schnorr identification protocol

SUMMARY. A proves its identity to B in a 3-pass protocol.

1. *Selection of system parameters.*
   - A suitable prime p is selected such that $p-1$ is divisible by another prime q. (Discrete logarithms modulo p must be computationally infeasible e.g., $p \approx 2^{1024}, q \geq 2^{160}$.)
   - An element β is chosen, $1 \leq \beta \leq p-1$, having multiplicative order q. (For example, for α generator mod $p, \beta = \alpha^{\frac{(p-1)}{q}}$ mod p.)
   - Each party obtains an authentic copy of the system parameters (p, q, β) and the verification function (public key) of the trusted party T, allowing verification of T's signatures $S_T(m)$ on messages m. (S_T involves a suitable known hash function prior to signing, and may be any signature mechanism.)
   - A parameter t (e.g.,$t \geq 40), 2^t < q$, is chosen (defining a security level 2^t).

Schnorr identification protocol

Algorithm 4 Schnorr identification protocol

```
2. Selection of per-user parameters.
    • Each entity $A$ is given a unique identity $I_A$.
    • $A$ chooses a private key $a, 0 \leq a \leq q-1$, and computes
      $v = \beta^{-a} \bmod p$.
    • $A$ identifies itself by conventional means (e.g., passport) to $T$,
      transfers $v$ to $T$ with integrity, and obtains a certificate
      $cert_A = (I_A, v, S_T(I_A, v))$ from $T$ binding $I_A$ with $v$.
3. Protocol messages. The protocol involves three messages.
    • $A \rightarrow B : cert_A,\ x = \beta^r \bmod p$      (2)
    • $A \leftarrow B : e$ ( where $1 \leq e \leq 2^t < q$)
    • $A \rightarrow B : y = ae + r \bmod q$
```

Schnorr identification protocol

Algorithm 4 Schnorr identification protocol

4. *Protocol actions.* A identifies itself to verifier B as follows.
   - A selects a random r(the commitment), $1 \leq r \leq q-1$, computes (the witness) $x = \beta^r \bmod p$, and sends (2) to B.
   - B authenticates A's public key v by verifying T's signature on $cert_A$, then sends to A a (never previously used) random e (the challenge), $1 \leq e \leq 2^t$.
   - A checks $1 \leq e \leq 2^t$ and sends B (the response) $y = ae + r \bmod q$.
   - B computes $z = \beta^y v^e \bmod p$, and accepts A's identity provided $z = x$.

Secret sharing

- *Secret sharing* refers to methods for distributing a secret among a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined together; individual shares are of no use on their own.
- Secret sharing was invented independently by Adi Shamir and George Blakley in 1979.

Shamir's Secret Sharing (SSS)

- Shamir's Secret Sharing (SSS) is used to secure a secret in a distributed way, most often to secure other encryption keys. The secret is split into multiple parts, called *shares.* These shares are used to reconstruct the original secret. To unlock the secret via Shamir's secret sharing, you need a minimum number of shares. This is called the *threshold,* and is used to denote the minimum number of shares needed to unlock the secret.

Shamir's Secret Sharing (SSS)

- *Problem:* Company XYZ needs to secure its vault's passcode. They could use something standard, such as AES, but what if the holder of the key is unavailable or dies? What if the key is compromised via a malicious hacker or the holder of the key turns rogue, and uses their power over the vault to their benefit?
- This is where SSS comes in. It can be used to encrypt the vault's passcode and generate a certain number of shares, where a certain number of shares can be allocated to each executive within Company XYZ. Now, only if they pool their shares then they can unlock the vault. The threshold can be appropriately set for the number of executives, so the vault is always able to be accessed by the authorized individuals. If one or two shares fall into the wrong hands, they couldn't open the passcode unless the other executives cooperated.

Shamir's (t, n) Threshold Scheme.

- Shamir proposed a (t, n) threshold scheme that splits a secret $s \in S$ into n shares, which are distributed to n users.
- Splitting is done by a dealer using an algorithm called *share generation algorithm.* The algorithm uses a polynomial $f(x)$ of degree $t-1$ to generate and distribute shares.
- The secret is reconstructed based on interpolating a polynomial using Lagrange interpolation, which is reconstructed by t users.
- The users combine their shares to reconstruct a polynomial $f'(x)$ of degree less than t using reconstruction algorithm. The algorithm inputs the user's identity i and their share v_i, which forms a point or an ordered pair (i, v_i) for all $i = 1, 2, \ldots, t$ and outputs the secret $f'(0) = s$.

Shamir's (t, n) Threshold Scheme.

Shamir's scheme has the following important properties.

- Share size is exactly equal to secret size.
- If a new player joins or leaves, it is easy to add or delete shares without affecting the other shares.
- It is easy to change the shares of the same secret just by changing the polynomial without breaching any security.
- $t - 1$ users do not reveal any information about the secret.

Lagrange interpolation.

Theorem

Given t distinct points (x_i, y_i) of the form $(x_i, f(x_i))$, where $f(x)$ is a polynomial of degree less than t, then $f(x)$ is determined by

$$f(x) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \tag{1}$$

Proof. Let $g(x)$ be the right side of (1). For each x_i, we verify directly that $f(x_i) = g(x_i)$, so that $f(x) - g(x)$ is divisible by $x - x_i$. It follows that

$$\prod_{i=1}^{t} (x - x_i) \mid (f(x) - g(x)) \tag{2}$$

but since $\deg(f(x) - g(x)) < t$, the only polynomial of this degree satisfying equation (2) is $f(x) - g(x) = 0$. ■

Shamir's scheme.

Shamir's scheme is defined for a secret $s \in \mathbb{Z}/p\mathbb{Z}$ with p prime, by setting $a_0 = s$, and choosing $a_1, \dots, a_{t-1}$ at random in $\mathbb{Z}/p\mathbb{Z}$. The trusted party computes $f(i)$, where

$$f(x) = \sum_{k=0}^{t-1} a_k x^k, \tag{3}$$

for all $1 \leq i \leq n$. The shares $(i, f(i))$ are distributed to the n distinct parties. Since the secret is the constant term $s = a_0 = f(0)$, the secret is recovered from any t shares $(i, f(i))$, for $I \subset \{1, \dots, n\}$ by

$$s = \sum_{i \in I} c_i f(i), \text{ where each } c_i = \prod_{\substack{j \in I \\ j \neq i}} \frac{j}{j - i}. \tag{4}$$

Shamir's scheme - example

Example. Shamir secret sharing with $p = 31$. Let the threshold be $t = 3$, and the secret be $7 \in \mathbb{Z}/31\mathbb{Z}$. We choose elements at random $a_1 = 19$ and $a_2 = 21$ in $\mathbb{Z}/31\mathbb{Z}$, and set $f(x) = 7 + 19x + 21x^2$. As the trusted part, we can now generate as many shares as we like

$$\begin{aligned}
(1, f(1)) &= (1, 16) & (5, f(5)) &= (5, 7)\\
(2, f(2)) &= (2, 5) & (6, f(6)) &= (6, 9)\\
(3, f(3)) &= (3, 5) & (7, f(7)) &= (7, 22)\\
(4, f(4)) &= (4, 16) & (8, f(8)) &= (8, 15)
\end{aligned}$$

which are distributed to the holders of the share recipients, and the original polynomial $f(x)$ is destroyed. The secret can be recovered from the formula

$$f(x) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \implies f(0) = \sum_{i=1}^{t} y_i \prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{x_j}{x_j - x_i} \qquad (5)$$

using any t shares $(x_1, y_1), \ldots, (x_t, y_t)$.

Shamir's scheme - example

If we take the first three shares (1,16), (2,5), (3,5), we compute

$$\begin{aligned} f(0) &= \frac{16\cdot 2\cdot 3}{(1-2)(1-3)} + \frac{5\cdot 1\cdot 3}{(2-1)(2-3)} + \frac{5\cdot 1\cdot 2}{(3-1)(3-2)} \\ &= 3\cdot 2^{-1} + 15\cdot(-1) + 10\cdot 2^{-1} = 17 - 15 + 5 = 7. \end{aligned}$$

This agrees with the same calculation for the shares (1,16), (5,7), and (7,22),

$$\begin{aligned} f(0) &= \frac{16\cdot 5\cdot 7}{(1-5)(1-7)} + \frac{7\cdot 1\cdot 7}{(5-1)(5-7)} + \frac{22\cdot 1\cdot 5}{(7-1)(7-5)} \\ &= 2\cdot 24^{-1} + 18\cdot(-8)^{-1} + 17\cdot 12^{-1} = 13 + 21 + 4 = 7. \end{aligned}$$