9.4.2 Blum-Goldwasser Cryptosystem

- Key generation:
  (i) \( p \neq q \) primes, \( p, q \equiv 3 \pmod{4}, n = p \cdot q \)
  (ii) compute \( a, b \) with \( a \cdot p + b \cdot q = 1 \)
  (iii) Public key \( n \), private key \( (p, q, a, b) \)

- Encryption: Let \( h = \lfloor \log_2 \log_2 (n) \rfloor \)
  Message: \( m_i = (m_{i1}, \ldots, m_{ib}) \in \{0, 1\}^{b \cdot b} \), each \( m_i \)
  \( b \) of size \( h \) (bits)

  **Blum-Blum-Shub (BBS) generator** for generating pseudo-random bits \( b_i^* \)
  - Select a random \( QR \) mod \( n \): \( x_0 \)
    (Select randomly \( x_0 \in \mathbb{Z}_n^* \), let \( x_0 = 2^t \) mod \( n \))
  - Iterate: \( t_i = t_{i-1}^2 \) mod \( n \), \( i = 1, \ldots, b \cdot h \)
    \( b_i^* \) denotes the \( k \) least significant (least) bits of \( t_i^* \)
    \( c_i = m_i \oplus b_i^* \)
  - Cipher text: \( c = (c_1, \ldots, c_{i-1}, t_i, t_{i+1}) \)

- Decryption:
  \( d_1 = \left( \frac{p+1}{4} \right)^{b+1} \) mod \( (p-1) \)
  \( d_2 = \left( \frac{q+1}{4} \right)^{b+1} \) mod \( (q-1) \)
  \( u = (t_i^{b+1})^{d_1} \) mod \( p \)
  \( v = (t_i^{b+1})^{d_2} \) mod \( q \)
  \( x_0 = (v \cdot a \cdot p + u \cdot b \cdot q) \) mod \( n \)
  - Iterate: \( t_i = t_{i-1}^2 \) mod \( n \), \( i = 1, \ldots, b+1 \)
    \( m_i = c_i \oplus b_i^* \)
Prop. 9.15 The decryption of the BC cryptosystem is correct.

Proof: The only remaining point is to show that $t_0$ is correct.

\[ x_i = 0, \ldots, x_l : x_i QR \mod n \Rightarrow x_i QR \mod p = x_i \equiv 1 \pmod p \]

Hence,

\[ x_{i+1} \equiv \left( x_i \right)^{p+1} \equiv x_i^{p+1} \equiv x_i \cdot x_i \equiv x_i \pmod p \]

By induction it follows:

\[ u = (x_{i+1})^{d_1} \equiv (x_{i+1})^t \equiv \left( \frac{p+1}{q} \right)^{\frac{d_1}{q}} \equiv \left( \frac{p+1}{q} \right)^{d_1} \equiv x_i \equiv t_0 \pmod p \]

\[ \exists \kappa \in \mathbb{Z} : d \equiv \kappa \pmod {p-1} \]

\[ x_0 \equiv x + \kappa (p-1) \]

\[ x_0 \equiv x + \kappa \pmod p \]

\[ x_0 \equiv x^{\frac{p-1}{k}} \equiv x \pmod {p-1} \]

Analogously,

\[ u \equiv x_i \pmod p \]

\[ x_0 \equiv n \cdot x \cdot p + n \cdot b \cdot q \pmod p \]

By Prop. 8.1

\[ x_0 \equiv n \cdot x \cdot p + n \cdot b \cdot q \pmod p \]

\[ x_0 \equiv n \cdot x \cdot p + n \cdot b \cdot q \pmod p \]
Example 9.15/ (with artificially small parameters)

Key generation: \( p = 499, \ q = 547 \) \( \equiv 3 \mod 4 \)

\[ n = p \cdot q = 272953 \] \( \log_2 (\log_2 (n)) \approx 4.4752 \)

\[ F_E: a \cdot p + b \cdot q = 7 = q \cdot d(p, q) \quad a = -57, \ b = 52 \]

Encryption: \( h = 9, \ t = 5 \)

\[ m = (m_1, \ldots, m_5) = (1001, 1100, 10001, 10000, 1100) \]

Choose random \( c \in Z_n^* \) : \( c = 399 \)

\[ x_0 = 399^2 \mod n = 159201 \]

\[
\begin{array}{c|c|c|c}
 i & x_i = x_{i-2}^2 \mod n & b_i & c_i = m_i \oplus b_i \\
 1 & 180539 & 1011 & 0010 \\
 2 & 193932 & 1100 & 0000 \\
 3 & 245613 & 1101 & 1100 \\
 4 & 120286 & 1110 & 1110 \\
 5 & 40632 & 1000 & 0100 \\
 6 = t + 1 & 139680 & & \\
\end{array}
\]

\( c = (0010, 10000, 1100, 1110, 0100, 139680) \)

Decryption:

\[ d_1 = \left(\frac{p+1}{4}\right) t + 1 \mod p - 1 = 463 \]

\[ d_2 = \left(\frac{q+1}{4}\right) t + 1 \mod (q - 1) = 337 \]

\[ u = (x_t \overline{u}) \mod p = 20 \]

\[ v = (x_t \overline{v}) \mod q = 24 \]

\[ x_0 = v \cdot a \cdot p + u \cdot b \cdot q \mod n = 159201 \]
Security of the BC cryptosystem

a) A eavesdropper sees the QR $X + 1$. To determine $X$, means to solve $QRSP(X+1, m)$, which is considered computationally infeasible.

b) The BC cryptosystem is vulnerable to chosen ciphertext attacks. Opponent access $X$. Opponent selects a random message $m \in \mathbb{Z}_n^*$, computes $X + 1 = m^2 \mod n$. There are 4 solutions $X + 1 = m^2 \mod n$.

If $X + 1 \neq \pm m$ then $gcd(X + 1, n) \in \mathbb{P}$.

If $X + 1 = \pm m$ then select a new random message $m$.

This attack is analogous to the one in the Rabin cryptosystem.

Efficiency of the BC system

a) The message expansion is constant by $\log_2^2(n)$ bits, if the representation of $X + 1$.

b) Computational effort is comparable to RSA, both in the encryption and decryption.
One-way hash functions: mapping messages of arbitrary length to a digest of fixed length \( m \), typically \( m = 64, 96, 160 \) bits.

- Signature schemes, sign the hash of a document rather than a long document itself.
- Data integrity (software protection, protection against viruses (MDC - Modification/Manipulation detection code, MAC - Message authentication code))

Hash functions are typically publicly known and involve no secret keys.

Formal description of hash functions:

- \( M \): message space (e.g., \( M = \{ 0,1 \}^* \))
- \( Y \): finite set of possible hash values (digest, hash digest, authentication tags) (e.g., \( Y = \{ 0,1 \}^n \); \( n \in \{64, 96, 160\} \))
- \( K \): key space (finite set)
- \( h \): hash function \( h : M \times K \rightarrow Y \); \( (m, k) \rightarrow h(m, k) \)

\( h \) is called \underline{unkeyed}, if \( |K| = 1 \) or \( h : M \rightarrow Y \)

\((m, h(m))\) is called a valid pair.
10.1 Security of hash functions

In the following we are considering unkeyed hash functions.

"It is computationally infeasible to compute preimages or to generate collisions." leads to the following:

Basic properties of cryptographic hash function $h : M \rightarrow Y$

1. Given $m \in M$, $h(m)$ is easy to compute.

Further, the solution of the following problems is computationally infeasible:

2. Given $y \in Y$, find $m \in M$ such that $h(m) = y$.

In this case $h$ is called one-way function or preimage resistant.

3. Given $m \in M$, find $m' \neq m$ such that $h(m') = h(m)$.

In this case $h$ is called second preimage resistant.

4. Find $m \neq m' \in M$ such that $h(m) = h(m')$.

In this case $h$ is called (strongly) collision free.

Note: Both $m$ and $m'$ may be freely chosen.