Low-cost Public-key Cryptography for M2M

Using the **Hydra** Scalable Cryptographic Coprocessors

Chen-Mou Cheng

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
ccheng@cc.ee.ntu.edu.tw

October 19, 2011

Outline

Project **Hydra**

- Aims to provide low-cost public-key cryptography for M2M using scalable coprocessors
- Supports many PKCs (hence a "hydra") via firmware
- Raises the abstraction level for M2M security engineering

Hydra staff (13)

- PI: 鄭振牟 (a.k.a. Doug)
- Co-PIs: 楊柏因 (中央研究院), 吳安宇 (a.k.a. Andy)
- Intel champions: Jesse Walker, 趙玫瑗
- Postdoc: Peter Schwabe (a.k.a. 樂岩)
- PhD students: 陳明興, 沈文中
- Engineer: 王伯文
- MS students: 施傑仁, 蘇冠羽, 蕭名群, 胡永波 (復旦大學)

Outline

What is the problem? Why is it hard?

- Typical M2M systems run for a long time without human intervention
- Deployment must not require complicated assembly
  - Pairing Bluetooth devices is an upper bound (not necessarily tight)
- Furthermore, many sensors are deployed in a way that makes them difficult to receive maintenance services by security experts
  - E.g., in Zigbee, sufficient to compromise one sensor because the whole group uses the same key
- Cost is always an issue
  - Need to include all kinds of costs: BoM, application development, deployment, operation, maintenance, ...

Outline

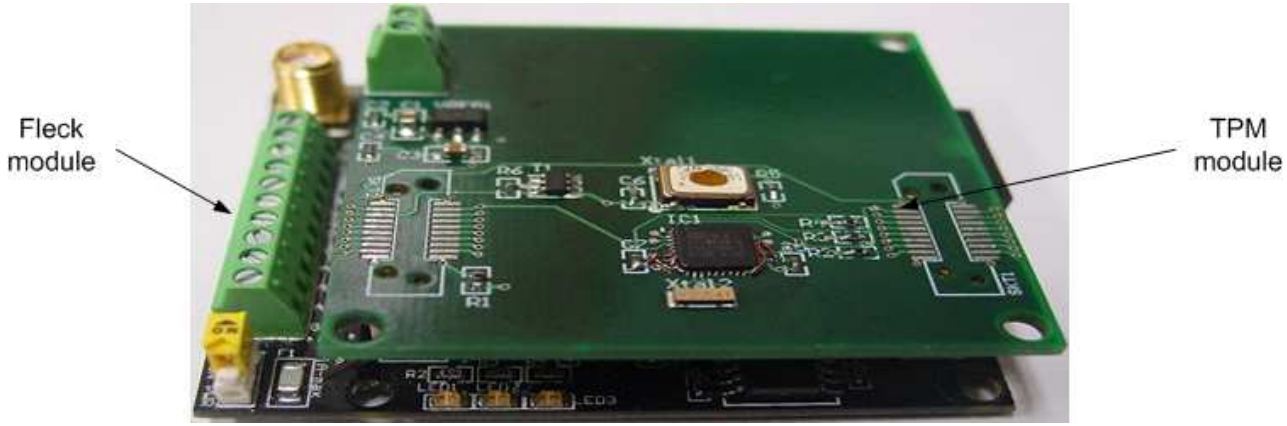
How is it done today? What are the limits of current practice?

- Firewalls provide network security by blocking malicious traffic
  - M2M network is decentralized $\Rightarrow$ "Firewall" model no longer works
- Little cryptography in deployed, real-world M2M systems
  - Cryptography often said to be too expensive to afford
- Use of software, symmetric-key cryptography
  - Many creative proposals from academia try to get around the obstacles by designing "light-weight" primitives and protocols
  - However, symmetric-key cryptography is becoming more "expensive" in terms of management and communication cost
- Use of software, asymmetric cryptography
  - Performance is "acceptable" but not good enough
  - Want to take this further by hardware acceleration

Example 1: CodeBlue (2004–)

- First, 'We are developing CodeBlue, an efficient wireless communication substrate for medical devices that addresses ad hoc network formation, naming and discovery, security and authentication, as well as filtration and aggregation of vital sign data. CodeBlue is designed to operate across a wide range of devices, including low-power "motes," PDAs, and PCs, and addresses the special robustness and security requirements of medical care settings.'
- Later, "An important shortcoming of the current CodeBlue prototype is its lack of security."
- Despite: "Aside from the obvious security considerations with sensitive patient data, United States law mandates that medical devices meet the privacy requirements of the 1996 Health Insurance Portability and Accountability Act (HIPAA). Recent work on private-key and public-key cryptography schemes for sensor net- works [29, 22, 38] is applicable here, but must be integrated into an appropriate authentication and authorization framework."

Example 2: secFleck (2010–)



- Idea: Augment sensors with TPMs to use PKC
  - Symmetric session key encryption/decryption
  - Sensor node symmetric session key request/assignment operation
  - Group key establishment operation
  - Secure software update protocol
    - ★ Similar to Hui and Culler's Deluge, a MOAP (Multihop Over the Air Programming) protocol
    - ★ Can verify the signature of the a 256-byte page in 59 ms, which is more than 4300 bytes/second, approximately 50 times faster than the average 88.4 bytes/second dissemination rate achieved by Deluge in a 75 node network

MIFARE Classic demo video

- http://www.youtube.com/watch?v=XpA8U-bwrM4

Outline

Information security

- "Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction." [W]
  - Wikipedia, the most reliable source of information in the world ;-)

CIA concepts

- Confidentiality
  - "...to prevent the disclosure of information to unauthorized individuals or systems" [W]
- Integrity
  - "...data cannot be modified undetectably" [W]
- Availability
  - "For any information system to serve its purpose, the information must be available when it is needed." [W]

More concepts

- Authentication
  - "It should be possible for the receiver of a message to ascertain its origin; an intruder should not be able to masquerade as someone else." [S]
    - ★ B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. Wiley. October, 1996.
- Non-repudiation
  - "...one party of a transaction cannot deny having received a transaction nor can the other party deny having sent a transaction" [W]

Related concepts

- Privacy
- Stenography
  - "...serves to hide secret messages in other messages" [S]
- Digital-right management
  - "...access control technologies that can be used by hardware manufacturers, publishers, copyright holders and individuals to limit the use of digital content and devices" [W]

Cryptography and cryptanalysis

- Cryptography "... is about communication in the presence of adversaries."
  - Ron Rivest, cryptographer and co-inventor of the RSA algorithm
- "Cryptanalysis ... is the practice of codebreaking or cracking the code."
  - Wikipedia, the most reliable source of information in the world ;-)

Cryptanalytic attacks

- Ciphertext-only attack
- Known-plaintext attack
- Chosen-plaintext attack
- Adaptive-chosen-plaintext attack
- Chosen-cihpertext attack
- Chosen-key attack
- Rubber-hose cryptanalysis

Classical ciphers and cryptanalysis

- Transposition ciphers
  - Caesar cipher: `YES` (plaintext) $\stackrel{5}{\Longrightarrow}$ `DJK` (ciphertext)
- Substitution ciphers
  - Monoalphabetic substitution ciphers
  - Polyalphabetic substitution ciphers
    - ★ Vigenère ciphers:
      `STEAKSTEAKSTE` (key)
      `attacktonight` (plaintext)
      `SMXAMCMSNSYAX` (ciphertext)
  - Polygraphic substitution ciphers
- Frequency analysis is the main cryptanalysis technique

Cryptography in early twentieth century

- One-time pads
  - Invented by Gilbert Vernam of AT&T in 1917
  - Later proved to be information-theoretically secure by Claude Shannon
- Rotor machines in World War II
  - Japan: Type-B cipher machine (紫密)
    - ★ 紫密的破解導致帝國海軍聯合艦隊司令長官元帥海軍大將正三位大 勛位功一級山本五十六的陣亡
  - Germany: Enigma machine
    - ★ Cryptanalyzed by Alan Turing using *bombe*, an electro-mechanical device that breaks Enigma via simulation and exhaustive search
    - ★ A generalization of *bomba kryptologiczna* (Polish for "cryptologic bomb"), designed by Polish cryptologist Marian Rejewski

Modern ciphers

- Ideal ciphers should behave like a *random permutation* on messages
  - Usually think of plaintext consisting of block of bits rather than letters
- The key decides the permutation in action

Cryptographic hash functions

- "... a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value" [W]
- Examples: MD5, SHA-1, SHA-2 (SHA-256/224, SHA-512/384)
- Can be used to construct message authentication codes
  - "... a one-way hash function with the addition of a secret key" [S]

Desired properties of hash functions

- Given a message m, it should be easy to compute its digest $H(m)$
- Preimage resistance
  - Given a digest h, it should be difficult to find any message m such that $h = H(m)$
- Second preimage resistance
  - Given a message m_1, it should be difficult to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$
- Collision resistance
  - It should be difficult to find two different messages $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$
  - Any $2n$-bit hash function can provide at most n-bit collision resistance due to Birthday bound

Passwords-based authentication

- ATMs/bank accounts
- Computers
- Web sites
- ...

Password weaknesses

- Small space
- Poorly chosen
- Reused
- Stored insecurely

Password weaknesses

- Small space
- Poorly chosen
- Reused
- Stored insecurely
- Humans!

Password attacks

- Personal information
- Weak storage
- Reusing passwords
- Dictionary

Password attacks

- Personal information
- Weak storage
- Reusing passwords
- Dictionary
- Rainbow table

Rainbow table

- "...a precomputed lookup table offering a time-memory tradeoff used in recovering the plaintext password from a password hash generated by a hash function"

Rainbow table

- "...a precomputed lookup table offering a time-memory tradeoff used in recovering the plaintext password from a password hash generated by a hash function"
- Possible defense mechanisms
  - Add iterations (increase time complexity)
  - Add salts (increase time and memory complexities)

Multiple iterations and salt

- Hash function: $h(v) \rightarrow R$
- Just using hash function is bad
  - Each $h(v_i)$ can be calculated quickly
  - Common $h(v_i)$'s can be precomputed and stored in a rainbow table
- Defense by adding iteration: hash many times
  $h'(v) = h(h(h(h(\cdots(h(v))\cdots) \rightarrow R'$
- Defense by adding salt: add a random and public salt each time
  $h'(v + s) \rightarrow R''$
- Typical recommendations
  - Number of iterations: ≥ 1000
  - Length of salt: ≥ 64 bits

Password-based key-derivation functions

- RFC 2898 (`http://www.ietf.org/rfc/rfc2898.txt`)
- PKCS #5 v2.1: Password-Based Cryptography Standard (`ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2_1.pdf`)

```
PBKDF2 (P, S, c, dkLen)

Options:        PRF         underlying pseudorandom function (hLen
                            denotes the length in octets of the
                            pseudorandom function output)

Input:          P           password, an octet string
                S           salt, an octet string
                c           iteration count, a positive integer
                dkLen       intended length in octets of the derived
                            key, a positive integer, at most
                            (2^32 - 1) * hLen

Output:         DK          derived key, a dkLen-octet string
```

Detail of PBKDF2

- $DK = T_1 || T_2 || \cdots || T_\ell$
  - $\ell = \lceil dkLen / hLen \rceil$
  - $T_i = F(P, S, c, i) = U_1 \oplus U_2 \oplus \cdots \oplus U_c$
    - $U_1 = PRF(P, S || INT(i))$
    - $U_2 = PRF(P, S || U_1)$
    - $\ldots$
    - $U_c = PRF(P, S || U_{c-1})$
- The default pseudorandom function is `HMAC-SHA-1`

Symmetric- vs. asymmetric-key cryptography

- Problem: Alice wants to send Bob a secret message through mail

Symmetric- vs. asymmetric-key cryptography

- Problem: Alice wants to send Bob a secret message through mail
- Symmetric-key cryptographer's solution
  - Alice puts the secret message in a box
  - Alice locks the box using a padlock to which she has a key
  - Alice sends the box to Bob through mail
  - Bob can now use an identical copy of Alice's key to open the box

Symmetric- vs. asymmetric-key cryptography

- Problem: Alice wants to send Bob a secret message through mail
- Symmetric-key cryptographer's solution
  - Alice puts the secret message in a box
  - Alice locks the box using a padlock to which she has a key
  - Alice sends the box to Bob through mail
  - Bob can now use an identical copy of Alice's key to open the box
    - ★ Key-exchange problem: How does Bob obtain the key?

Symmetric- vs. asymmetric-key cryptography

- Problem: Alice wants to send Bob a secret message through mail
- Symmetric-key cryptographer's solution
  - Alice puts the secret message in a box
  - Alice locks the box using a padlock to which she has a key
  - Alice sends the box to Bob through mail
  - Bob can now use an identical copy of Alice's key to open the box
    - ★ Key-exchange problem: How does Bob obtain the key?
- Asymmetric-key cryptographer's solution
  - Bob and Alice have separate padlocks
  - Alice asks Bob to send his open padlock to her through mail
  - Alice uses it to lock a box containing her message
  - Alice sends the locked box to Bob
  - Bob can now unlock the box with his own key

Symmetric- vs. asymmetric-key cryptography

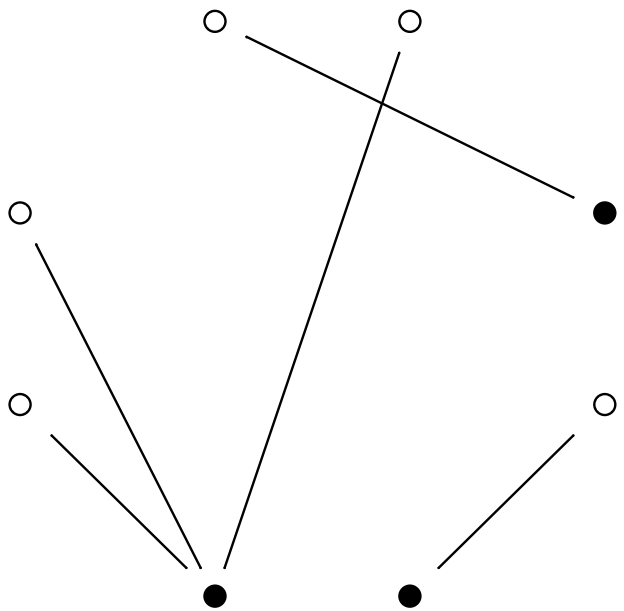
- Problem: Alice wants to send Bob a secret message through mail
- Symmetric-key cryptographer's solution
  - Alice puts the secret message in a box
  - Alice locks the box using a padlock to which she has a key
  - Alice sends the box to Bob through mail
  - Bob can now use an identical copy of Alice's key to open the box
    - ★ Key-exchange problem: How does Bob obtain the key?
- Asymmetric-key cryptographer's solution
  - Bob and Alice have separate padlocks
  - Alice asks Bob to send his open padlock to her through mail
  - Alice uses it to lock a box containing her message
  - Alice sends the locked box to Bob
  - Bob can now unlock the box with his own key
    - ★ Bob's key never leaves his hand!

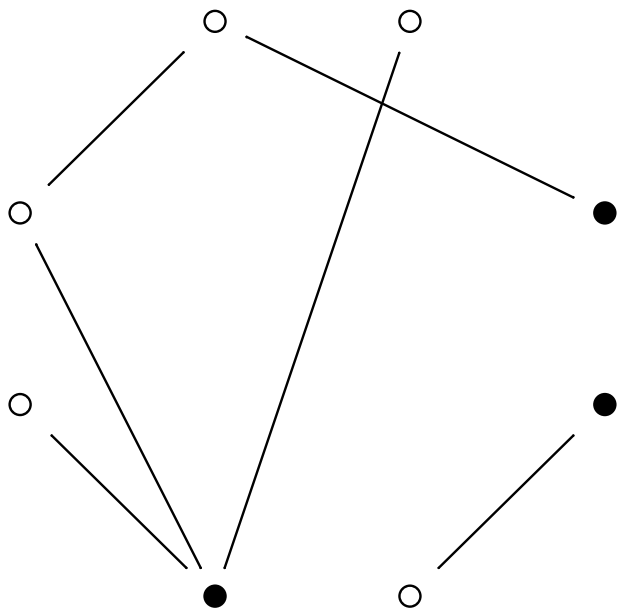
Perfect codes

- Some graph-theory terminology
  - A *graph* is a collection of *vertices* and *edges*
  - An edge connects two vertices
  - The *neighborhood* of a vertex consists of the vertex itself and all vertices that are joined to it by an edge

Perfect codes

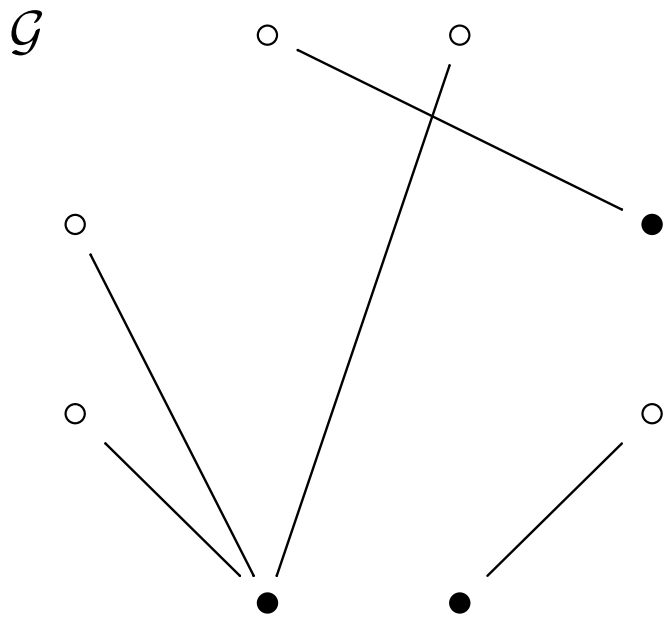
- Some graph-theory terminology
  - A *graph* is a collection of *vertices* and *edges*
  - An edge connects two vertices
  - The *neighborhood* of a vertex consists of the vertex itself and all vertices that are joined to it by an edge
- A *perfect code* in a graph is a subset of vertices such that every vertex is in the neighborhood of one and only one vertex in the subset
  - Not all graphs have perfect codes
  - The problem of finding a perfect code in a graph is NP-hard





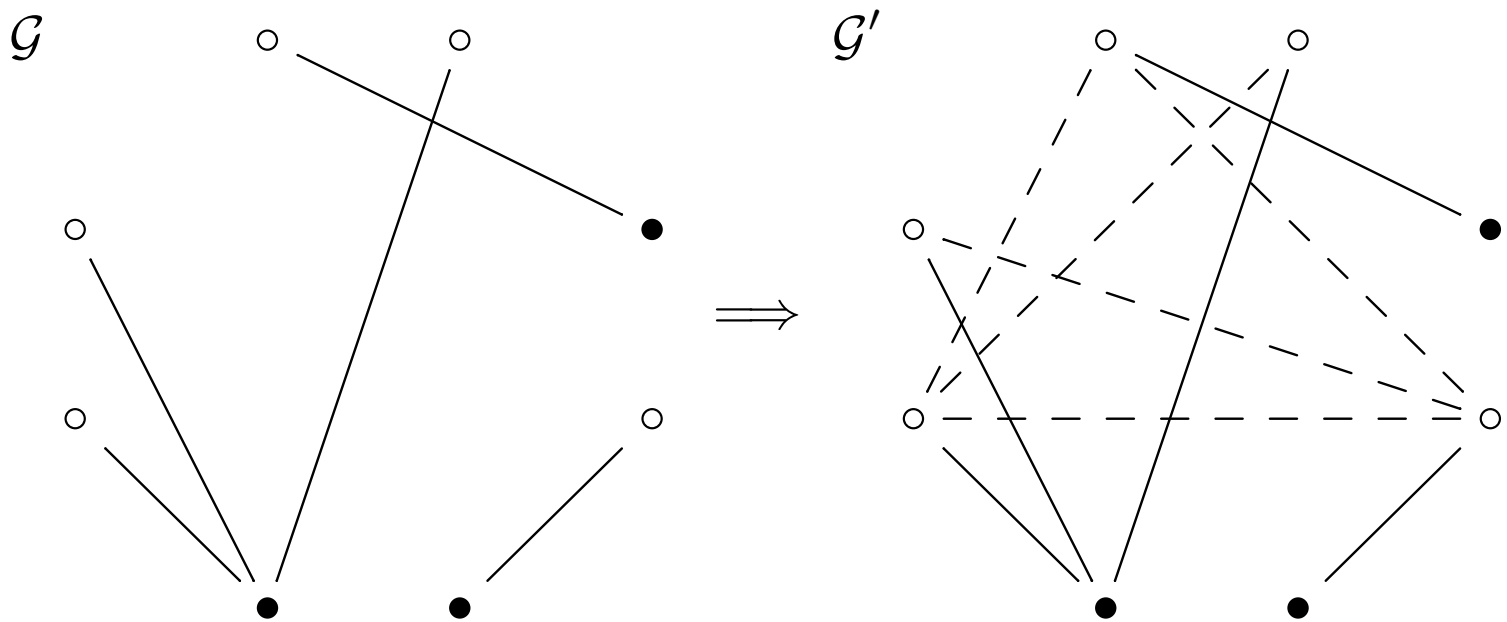
A toy public-key cryptosystem based on perfect codes

- Key generation
  - Start with a simple graph $\mathcal{G}$ containing a perfect code $\mathbb{C}$



A toy public-key cryptosystem based on perfect codes

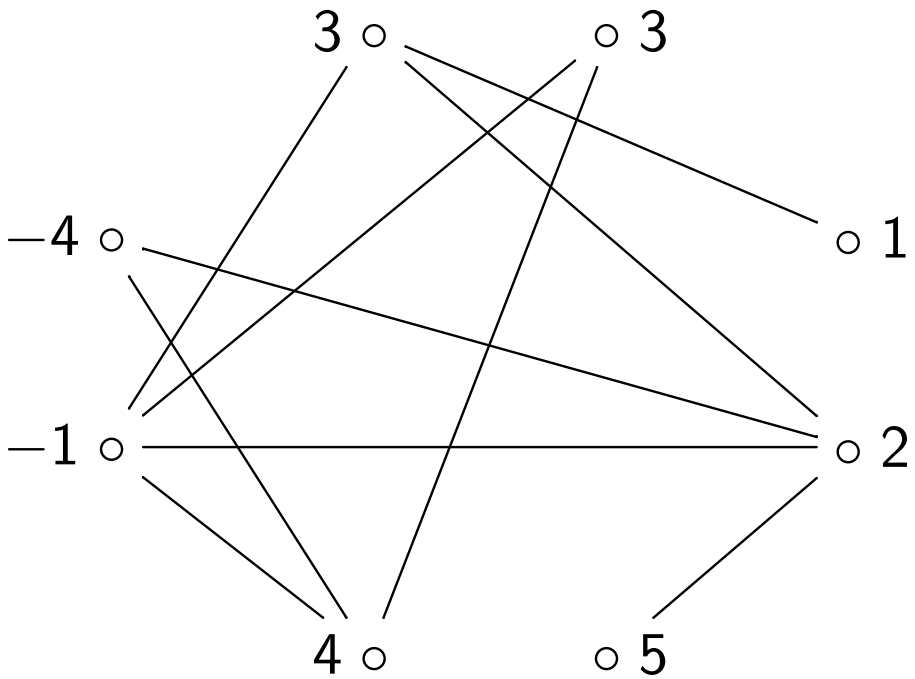
- Key generation
  - Start with a simple graph $\mathcal{G}$ containing a perfect code $\mathbb{C}$



  - Add to $\mathcal{G}$ new edges, making sure they do not touch vertices in $\mathbb{C}$
- Public key: the resulting graph $\mathcal{G}'$
- Private key: the perfect code $\mathbb{C}$

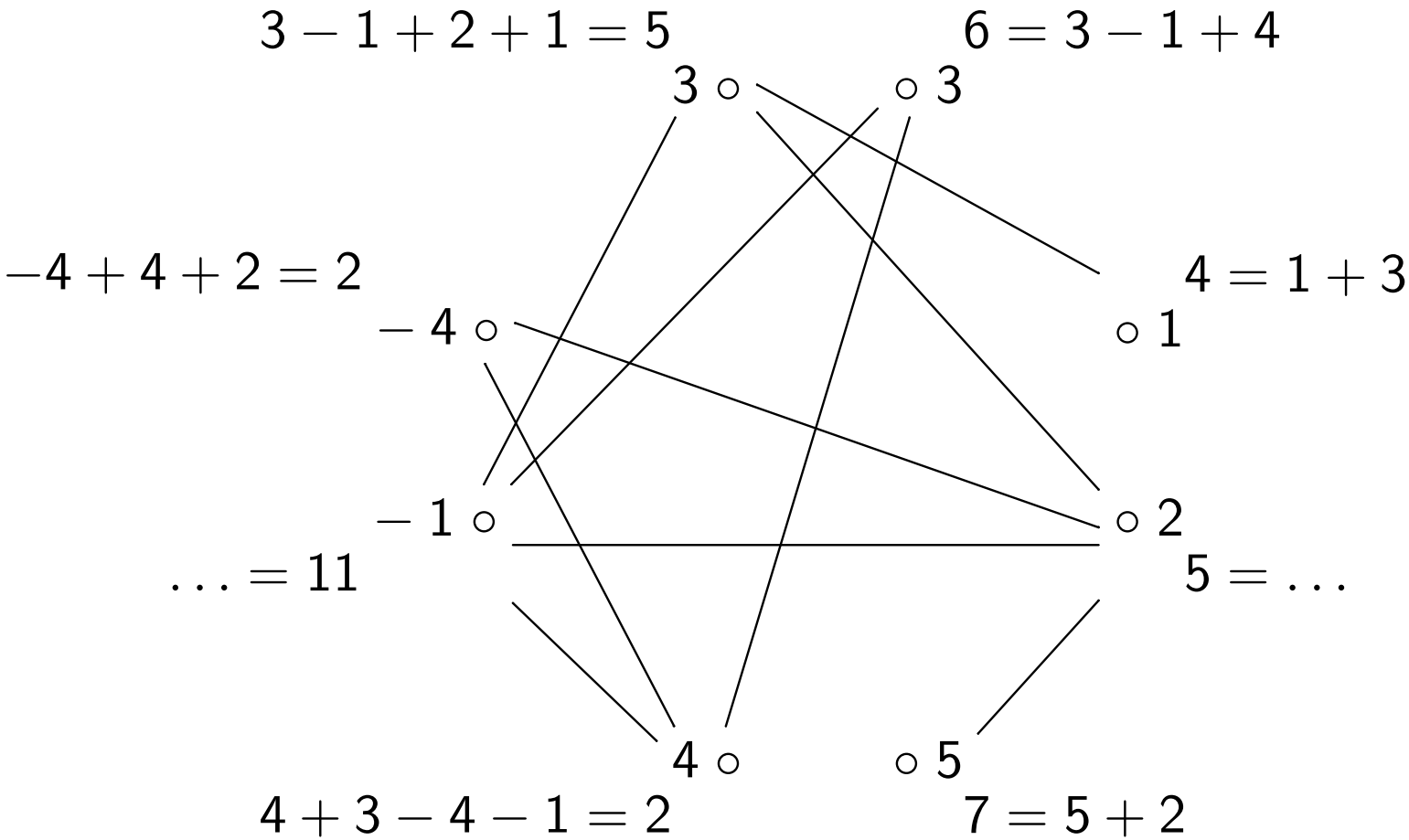
First step of encryption

- To encrypt an integer, first partition it into eight shares
- Example: $13 = 1 + 3 + 3 - 4 - 1 + 4 + 5 + 2$

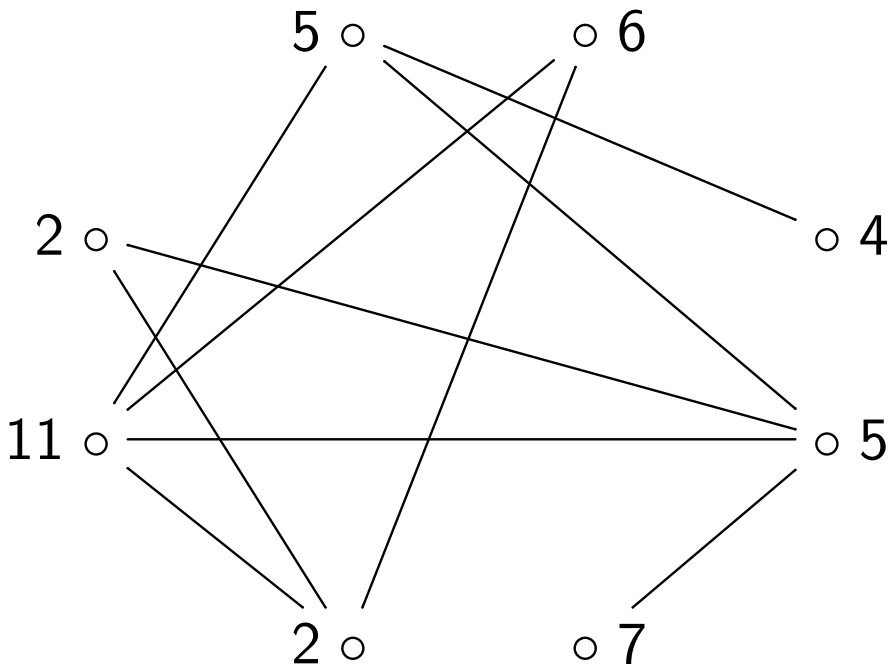


Second step of encryption

- Each node sums up all the values on the nodes in its neighborhood

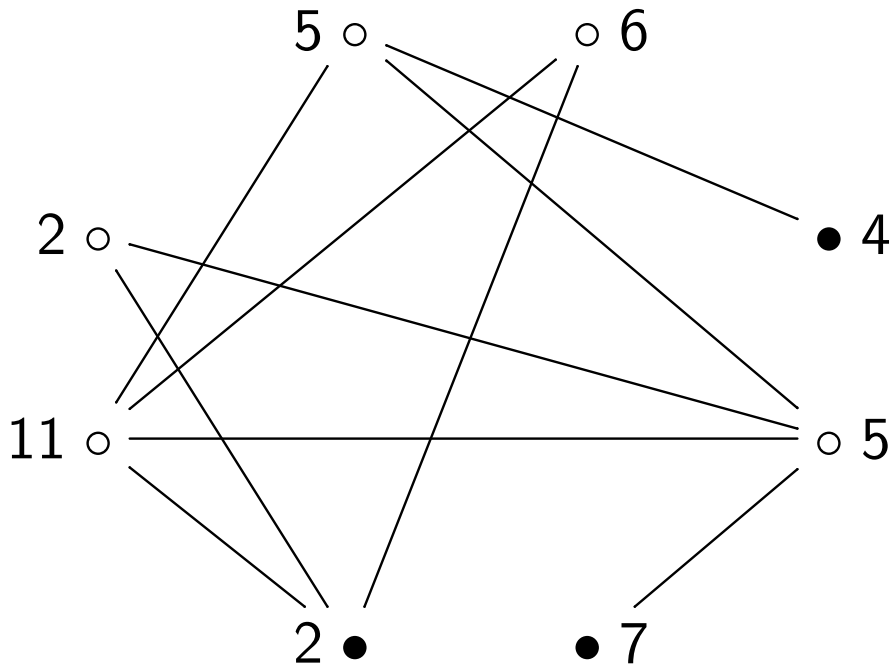


Ciphertext



Decryption

- Simply sum up all the values on the nodes in $\mathbb{C}$
- $4 + 2 + 7 = 13$



Why does it work?

Why does it work?

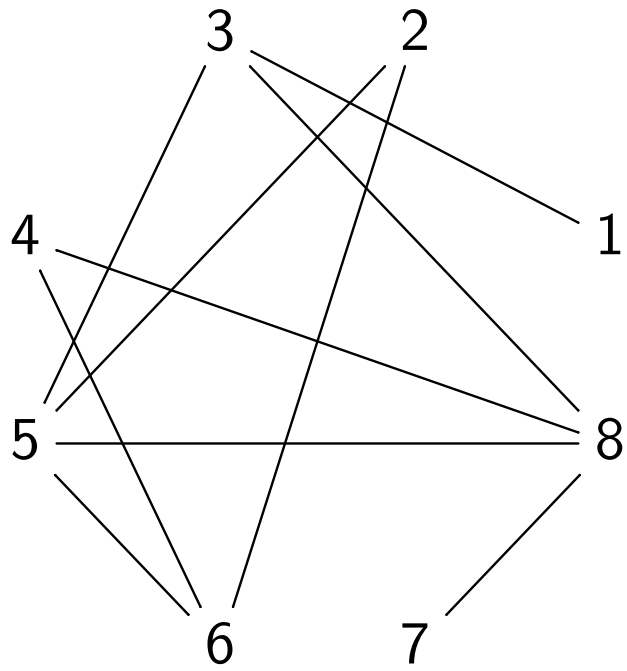
- Because each node will send its value to *exactly* one node in $\mathbb{C}$

How to break it?

How to break it?

- Write down the *adjacency matrix* of $\mathcal{G}'$:

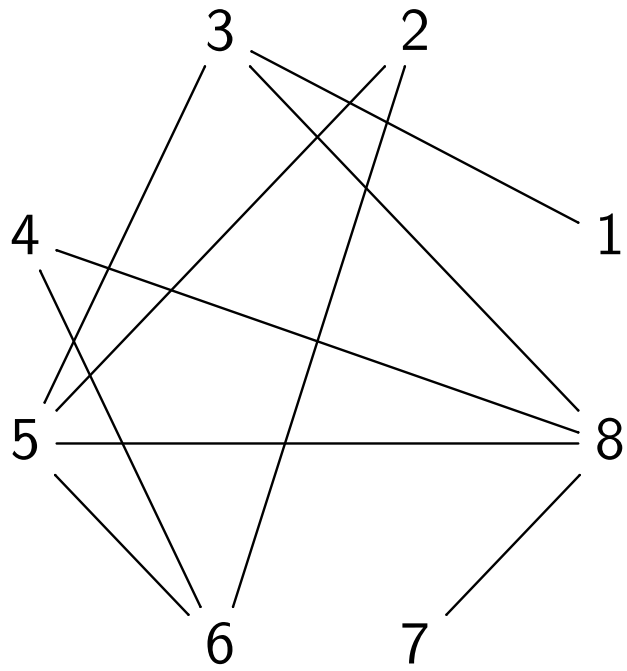
$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$



How to break it?

- Write down the *adjacency matrix* of $\mathcal{G}'$:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

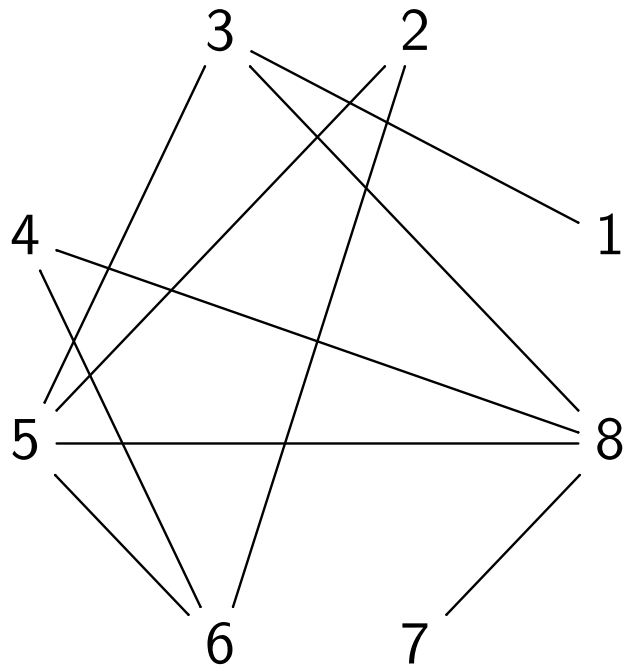


- Write the ciphertext as a vector $\mathbf{y} = (4, 6, 5, 2, 11, 2, 7, 5)$

How to break it?

- Write down the *adjacency matrix* of $\mathcal{G}'$:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$



- Write the ciphertext as a vector $\mathbf{y} = (4, 6, 5, 2, 11, 2, 7, 5)$
- Solve $A\mathbf{x} = \mathbf{y}$ using linear algebra!

Example: Rabin cryptosystem

- "... Rabin system is 'more secure' in this sense than is RSA, and will remain so until a general solution for the factorization problem is discovered, or until the RSA problem is discovered to be equivalent to factorization" [W]
- Key generation
  - "Choose two large distinct primes p and q. One may choose $p \equiv q \equiv 3 \pmod 4$ to simplify the computation of square roots modulo p and q"
  - "Let $n = pq$. Then n is the public key. The primes p and q are the private key."
- Encryption
  - $c = m^2 \bmod n$
- Decryption
  - $m_p = \sqrt{c} \bmod p = c^{(p+1)/4} \bmod p$
  - $m_q = \sqrt{c} \bmod q = c^{(q+1)/4} \bmod q$
  - Recover m via the Chinese remainder theorem

Digital signature

- Example: "Signing Documents with Public-Key Cryptography and One-Way Hash Functions" [S]
  1. Alice produces a one-way hash of a document.
  2. Alice encrypts the hash with her private key, thereby signing the document.
  3. Alice sends the document and the signed hash value.
  4. Bob produces a one-way hash of the document that Alice sent. He then, using the digital signature algorithm, decrypts the signed hash with Alice's public key. If the signed hash matches the hash he generated, the signature is valid.

Key exchange protocols

- Example: "Kerberos," a variant of Needham-Schroeder [S]
  1. Alice sends a message to Trent with her identity and Bob's identity.

$$A, B$$

  2. Trent generates a message with a timestamp, a lifetime, L, a random session key, and Alice's identity. He encrypts this in the key he shares with Bob. Then he takes the timestamp, the lifetime, the session key, and Bob's identity, and encrypts these in the key he shares with Alice. He sends both encrypted messages to Alice.

$$E_A(T, L, K, B), E_B(T, L, K, A)$$

  3. Alice generates a message with her identity and the timestamp, encrypts it in K, and sends it to Bob. Alice also sends Bob the message encrypted in Bob's key from Trent.

$$E_K(A, T), E_B(T, L, K, A)$$

  4. Bob creates a message consisting of the timestamp plus one, encrypts it in K, and sends it to Alice.

$$E_K(T + 1)$$

Key Exchange with Public-Key Cryptography [S]

1. Alice gets Bob's public key from KDC.
2. Alice generates a random session key, encrypts it using Bob's public key, and sends it to Bob.
3. Bob then decrypts Alice's message using his private key.
4. Both of them encrypt their communications using the same session key.

Man-in-the-Middle Attack [S]

1. Alice sends Bob her public key. Mallory intercepts this key and sends Bob his own public key.
2. Bob sends Alice his public key. Mallory intercepts this key and sends Alice his own public key.
3. When Alice sends a message to Bob, encrypted in "Bob's" public key, Mallory intercepts it. Since the message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Bob's public key, and sends it on to Bob.
4. When Bob sends a message to Alice, encrypted in "Alice's" public key, Mallory intercepts it. Since the message is really encrypted with his own public key, he decrypts it with his private key, re-encrypt it with Alice's public key, and sends it on to Alice.

Rivest and Shamir's Interlock Protocol [S]

1. Alice sends Bob her public key.
2. Bob sends Alice his public key.
3. Alice encrypts her message using Bob's public key. She sends half of the encrypted message to Bob.
4. Bob encrypts his message using Alice's public key. He sends half of the encrypted message to Alice.
5. Alice sends the other half of her encrypted message to Bob.
6. Bob puts the two halves of Alice's message together and decrypts it with his private key. Bob sends the other half of his encrypted message to Alice.
7. Alice puts the two halves of Bob's message together and decrypts it with her private key.

Diffie-Hellman [S]

1. Alice chooses a random large integer x and sends Bob
$$X = g^x \bmod n$$
2. Bob chooses a random large integer y and sends Alice
$$Y = g^y \bmod n$$
3. Alice computes
$$k = Y^x \bmod n$$
4. Bob computes
$$k = X^y \bmod n$$

Bellovin and Merritt's Encrypted Key Exchange (1/2) [S]

1. Alice generates a random public-key/private-key key pair. She encrypts the public key, K', using a symmetric algorithm and P as the key: $E_P(K')$. She sends Bob

$$A, E_P(K')$$

2. Bob knows P. He decrypts the message to obtain K'. Then, he generates a random session key, K, and encrypts with the public key he received from Alice and P as the key. He sends Alice

$$E_P(E_{K'}(K))$$

3. Alice decrypts the message to obtain K. She generates a random string, R_A, encrypts it with K, and sends Bob

$$E_K(R_A)$$

Bellovin and Merritt's Encrypted Key Exchange (2/2) [S]

4. Bob decrypts the message to obtain R_A. He generates another random string, R_B, encrypts both strings with K, and sends Alice the result.

$$E_K(R_A, R_B)$$

5. Alice decrypts the message to obtain R_A and R_B. Assuming the R_A she received from Bob is the same as the one she sent to Bob in step (3), she encrypts R_B with K and sends it to Bob.

$$E_K(R_B)$$

6. Bob decrypts the message to obtain R_B. Assuming the R_B he received from Alice is the same one as he sent to Alice in step (4), the protocol is complete. Both parties now communicate using K as the session key.

Zero-knowledge proof

- Example based on "Graph Isomorphism" [S]
- "Assume that Peggy knows the isomorphism between the two graphs, G_1 and G_2. The following protocol will convince Victor of Peggy's knowledge:"
  1. Peggy randomly permutes G_1 to produce another graph, H, that is isomorphic to G_1. Because Peggy knows the isomorphism between H and G_1, she also knows the isomorphism between H and G_2. For anyone else, finding an isomorphism between G_1 and H or between G_2 and H is just as hard as finding an isomorphism between G_1 and G_2.
  2. Peggy sends H to Victor.
  3. Victor asks Peggy either to:
     1. prove that H and G_1 are isomorphic, or
     2. prove that H and G_2 are isomorphic.
  4. Peggy complies. She either:
     1. proves that H and G_1 are isomorphic, without proving that H and G_2 are isomorphic, or
     2. proves that H and G_2 are isomorphic, without proving that H and G_1 are isomorphic.
  5. Peggy and Victor repeats steps (1) through (4) n times.

Outline

What are we trying to do?

- Secure M2M applications with hardware-assisted PKC
  - Multi-way authentication
  - Key exchange/distribution/management
  - Digital signature
  - Many advanced protocols and services can be built on top
    - ★ Example: Privacy-preserving discovery
- "Cheap" PKC
  - Hardware acceleration of core computations
  - Customizable for multiple vertical markets, allowing cost sharing
- "Future-proof" PKC
  - Algorithm agility, allowing "BIOS upgrades"
  - Post-quantum cryptography that resists emerging attacks
- "Management-free" cryptography
  - Lower total cost of ownership via PKC
  - Extreme: Identity-based cryptography $\Rightarrow$ No more PKI!

What's new in our approach, and why do we think it will be successful?

- To design and build **Hydra**: A proof-of-concept system to showcase the benefits of securing M2M sensors with hardware-assisted PKC
- Hardware and software components of **Hydra**
  - Scalable, programmable cryptographic coprocessors
  - Accompanying toolchains and software libraries
  - API to raise abstraction level for developing secured M2M applications
- All in all, computation is getting cheaper compared with other costs like management, communication, etc $\Rightarrow$ Time to base M2M security on strong (but computationally expensive) PKC!
- Initial success: TTS using 17K gates and 21 μA per signature
  - Much cheaper than exchanging, e.g., hash messages

Thanks!

- Questions or comments?

Thanks!

- Questions or comments?
- We're recruiting
  - Contact `doug@crypto.tw` if you're interested in building a more secure M2M world!