



# WK-08: Security in Wireless Broadcast

Never Stand Still

COMP4337/9337

Professor Sanjay K. Jha

School of Computer Science and Engineering, UNSW

# Outline

- Quick overview of Elliptic Curve Diffie-Hellman (ECDH) and Datagram TLS as lightweight solution for many wireless (IoT) solutions.
- Security Challenges in wireless broadcast
- Advanced techniques using hash-chains, Merkle Tree
- Application case study of Code dissemination in a multi-hop wireless network

# Datagram TLS (DTLS)

- SSL Designed to run on top of TCP
- Datagram TLS developed later to run over connectionless UDP
  - RFC 4347 for details
- Already supported by several implementations
- Very similar to TLS
  - Needs extra control messages as UDP doesn't provide these like TCP
  - Sequence number in record header to protect from Replay attack
- If very lossy network, may have issues with lot of retransmissions for reliability

-

# Elliptic Curve (ECC) Scheme

椭圆曲线加密算法也是用来获得对称秘钥的

- Key Agreement, aka, Elliptic Curve Diffie-Hellman (ECDH)
  - Allows for establishment of shared secret similar to DH
  - The shared key is then used for symmetric encryption or for further session/temporal key derivation
- Digital Signature: Elliptic Curve Digital Signature Algorithm (ECDSA), allows use of public/private key for signing a message and verification of signature, more efficient than RSA based DSA.

# Elliptic Curve Cryptography

- Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite field.
- ECC presents various benefits over RSA such as:
  - fast computation
  - small key size
  - compact signatures.
- For example, to provide equivalent security to 1024-bit RSA, an ECC scheme only needs 160 bits.

# Elliptic Curve Diffie-Hellman Key Exchange

第一步对用 elliptic curve 达成共识就是选择一个 curve 函数

1. Alice and Bob publicly agree on an elliptic curve  $E$  over a finite field  $\mathbb{Z}_p$ .
  2. 选择一个起始点  $B$
  3. 随机选择一个  $n$ , 通过打点计算的到  $p$ ,  
Alice chooses a random integer  $1 < \alpha < |E|$ , computes  $P = \alpha B$ , and sends  $P$  to Bob.  
*Alice keeps her choice of  $\alpha$  secret.*
  4. Bob chooses a random integer  $1 < \beta < |E|$ , computes  $Q = \beta B$ , and sends  $Q$  to Alice.  
*Bob keeps his choice of  $\beta$  secret.*
1. Alice and Bob choose  $E$  to be the curve  $y^2 = x^3 + x + 6$ .
  2. Alice and Bob choose the public base point to be  $B = (2, 4)$ .
  3. Alice chooses  $\alpha = 4$ , computes  $P = \alpha B = 4(2, 4) = (6, 2)$ , and sends  $P$  to Bob.  
*Alice keeps  $\alpha$  secret.*
  4. Bob chooses  $\beta = 5$ , computes  $Q = \beta B = 5(2, 4) = (1, 6)$ , and sends  $Q$  to Alice.  
*Bob keeps  $\beta$  secret.*

Maths not examinable



UNSW  
AUSTRALIA

# ECDH Key Exchange (cont.)

- |   |  |
|---|--|
| 5. Alice computes<br>$K_A = \alpha Q = \alpha(\beta B)$ . | 5. Alice computes<br>$K_A = \alpha Q = 4(1,6) = (4,2)$ . |
| 5. Bob computes<br>$K_B = \beta P = \beta(\alpha B)$ .    | 6. Bob computes<br>$K_B = \beta P = 5(6,2) = (4,2)$ .    |
| 7. The shared secret key is<br>$K = K_A = K_B$ .          | 7. The shared secret key is<br>$K = (4,2)$ .             |
- Even if Eve knows the base point B, or P or Q, she will not be able to figure out  $\alpha$  or  $\beta$ , so K remains secret!



**UNSW**  
AUSTRALIA

# Recap: How to use keys?

- Rule of thumb:
- Public Key Cryptography: slow
- Symmetric Cryptography: fast
- Hence, do not encrypt large messages with Public Key Cryptography
- Encrypt a random, fresh symmetric key with Public Key Cryptography
- Use this key and symmetric encryption to encrypt a large message
- For signature, only sign the hash value of messages
- Send the encrypted key, signature, and symmetrically encrypted message to your communication partner

-

# Challenges for Broadcast Security

- Broadcast applications need security
  - Packet injection or eavesdropping is easy
- Security solutions for point-to-point communication not suitable secure for broadcast
- Broadcast challenges
  - Scale to large audiences
  - Dynamic membership
  - Low overhead (computation & communication)
  - Packet loss
    - How to achieve reliability in broadcasts?

-

# Scale & Dynamics

- Small groups contain up to ~100 members
- Medium-size groups contain 100-1000 members
- Large groups contain 1000- $10^9$  members – e.g. IoT
- How does scale affect security?
- Dynamic membership: members may join and leave at any time
- How do dynamics affect security?

# Communication Pattern

- Group can be single-source broadcast
  - One-to-many
  - SSM: Single-source multicast, source-specific multicast  
多点传送
- Multiple-source broadcast
  - Some-to-many
- All members broadcast
  - Many-to-many

# Reliable Broadcast Transmission

- How to reliably and scalably disseminate data to large numbers of receivers? 散布传播
- Challenges
  - Ack implosion problem if receivers return Ack to sender for received packets
  - Nack implosion problem is severe as well
  - For large numbers of receivers, there usually is a fraction of them that do not obtain message
  - Local repair mechanisms (create tree topology and ask upstream parent for packet) faced numerous scalability difficulties
    - Accumulate ack (delayed) and send to parent node on the tree.

# End-to-End approach ?

- Trusted server authenticates each node and distributes the key.
- Use well-known E2E protocols such as (D)TLS, IPSEC, SSL.

## Pros

- Higher security level
- Dynamic grouping

## Cons

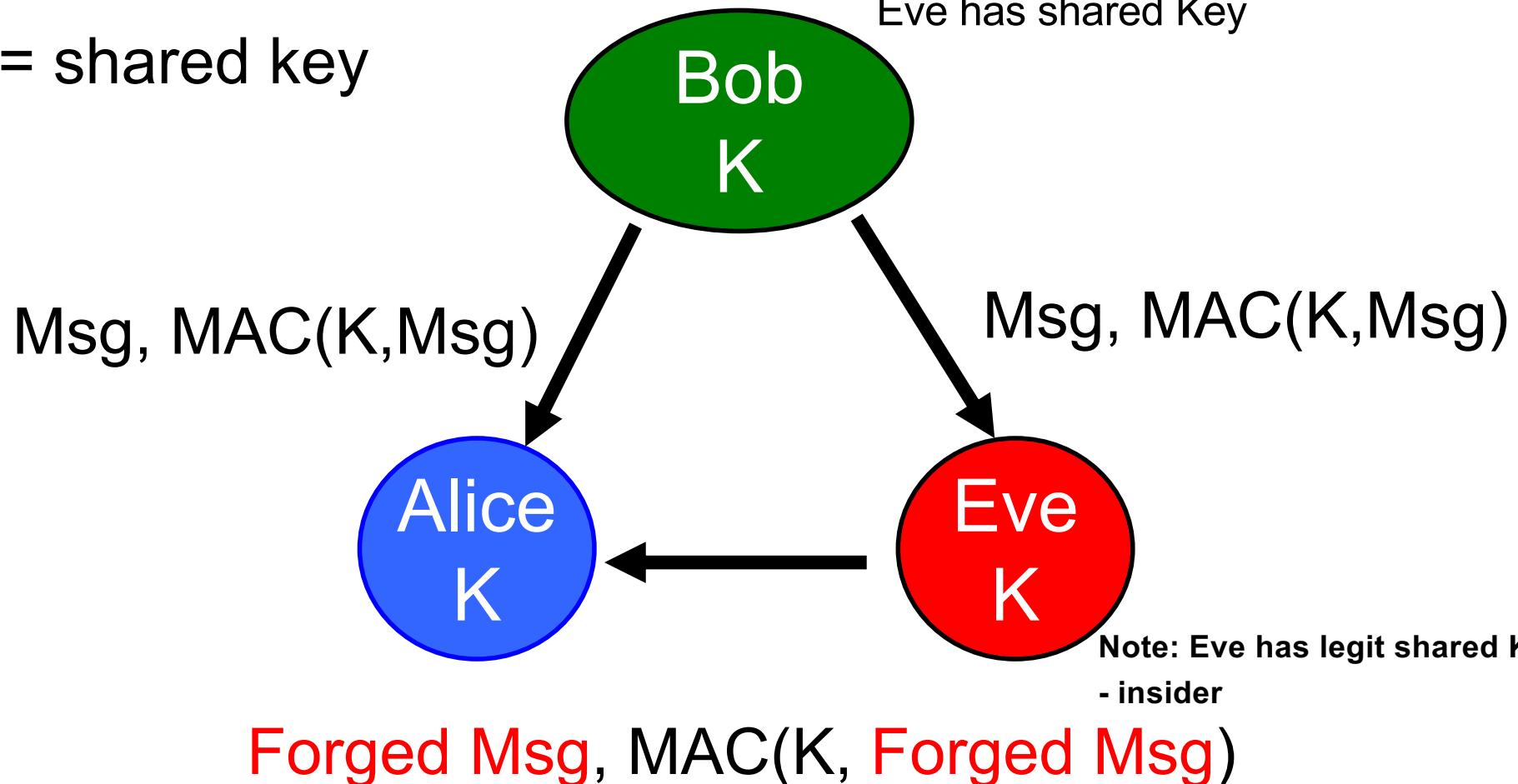
- Extremely higher ciphertext overhead due to one by one transfer
- Higher computation requirements

-

# Shared Key: Easy to Forge

$K$  = shared key

MAC: Message Authentication Code  
Eve has shared Key



# Asymmetric Key: Digital Signature

- Sign each packet and verify using Asymmetric key
- However, Signatures are expensive esp. for low end processors, e.g., RSA 2048:
  - High generation cost (~1 millisecond), High verification cost (~0.1 millisecond), High communication cost (256 bytes/packet)
- If we use one signature over multiple packets, intolerant to packet loss

# Trivial broadcast key distribution

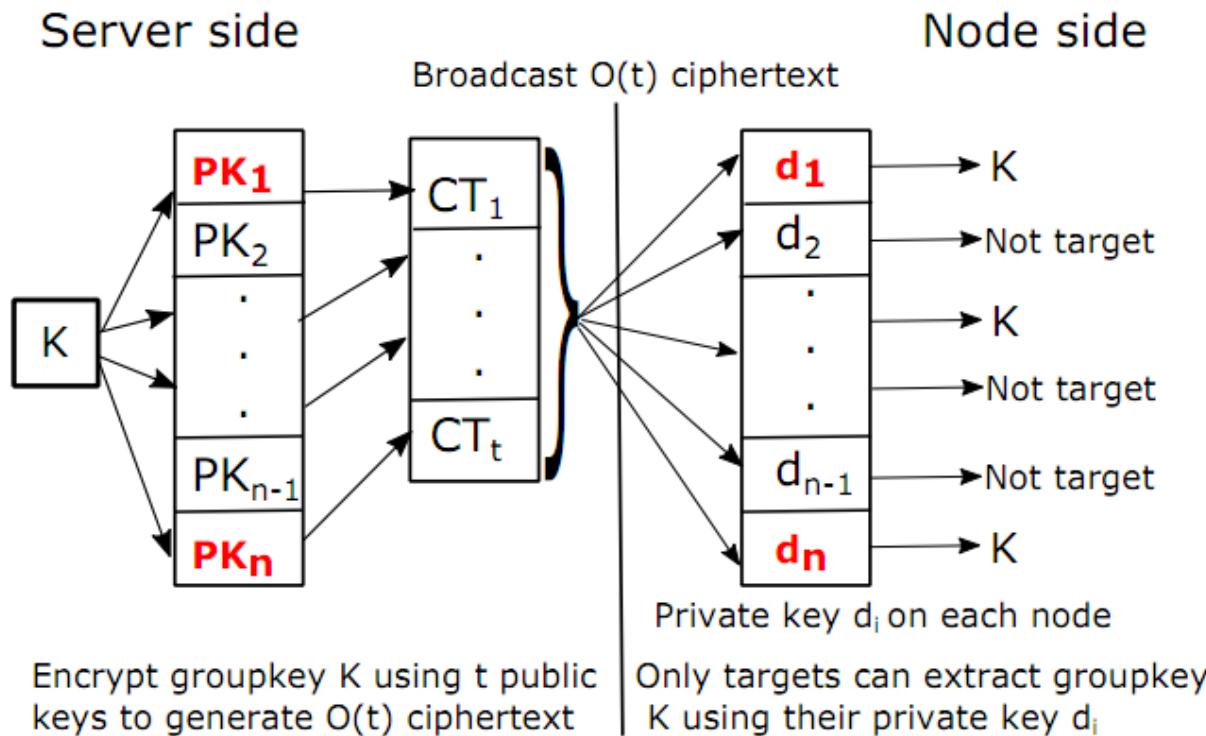


Figure 1: Trivial Broadcast Encryption scheme in an  $n$  nodes network targeting  $t$  nodes.  $K$ : shared group key.  $PK$ : Public Key.  $CT$ : Ciphertext.  $d$ : Decryption using each node's private key.



# Trivial (2)

- Very simple and effective system. Easy to implement.
- The only issue is linear ciphertext  $O(t)$  for key distribution.
- Good for applications with smaller number of target nodes.
- Possible solution for non-sensor applications such as Desktop, smart-phone
- Optional reading: Stinson, Douglas R. *Cryptography: theory and practice*. CRC press, 2005.

# Shamir's Secret Sharing Schemes

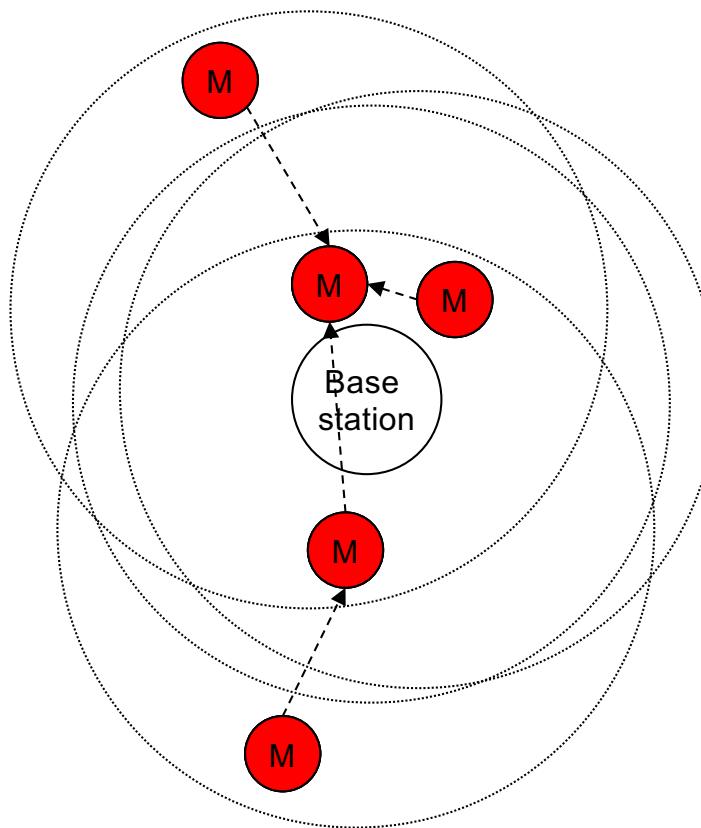
- Split secret into multiple parts aka shares
- **Threshold:** a minimum amount of shares needed to unlock the secret/vault.
  - Example: To access an important resource, a key can be generated by use of shares from various participants. Even if some shares get compromised, they can't Secret.
- Maths beyond scope: e.g. parabola uniquely described with 3 points,
  - multiple points (shares) can be distributed, threshold = 3 to define a parabola uniquely and find the secret S (e.g coefficient  $a_0$  of the polynomial )

# Hash Chain -basics

- Client generate 1000 hashes for password
  - Suggested by Lamport for password protection
- Server stores  $H^{1000}(\text{password})$
- Client willing to authenticate sends  $H^{999}(\text{password})$
- Server computes  $H^{1000}(\text{password}) = H(H^{999}(\text{password}))$ 
  - Match found, store  $H^{999}(\text{password})$  for next time
- Eavesdropper can't use  $H^{999}(\text{password})$  since server expects  $H^{998}(\text{password})$

-

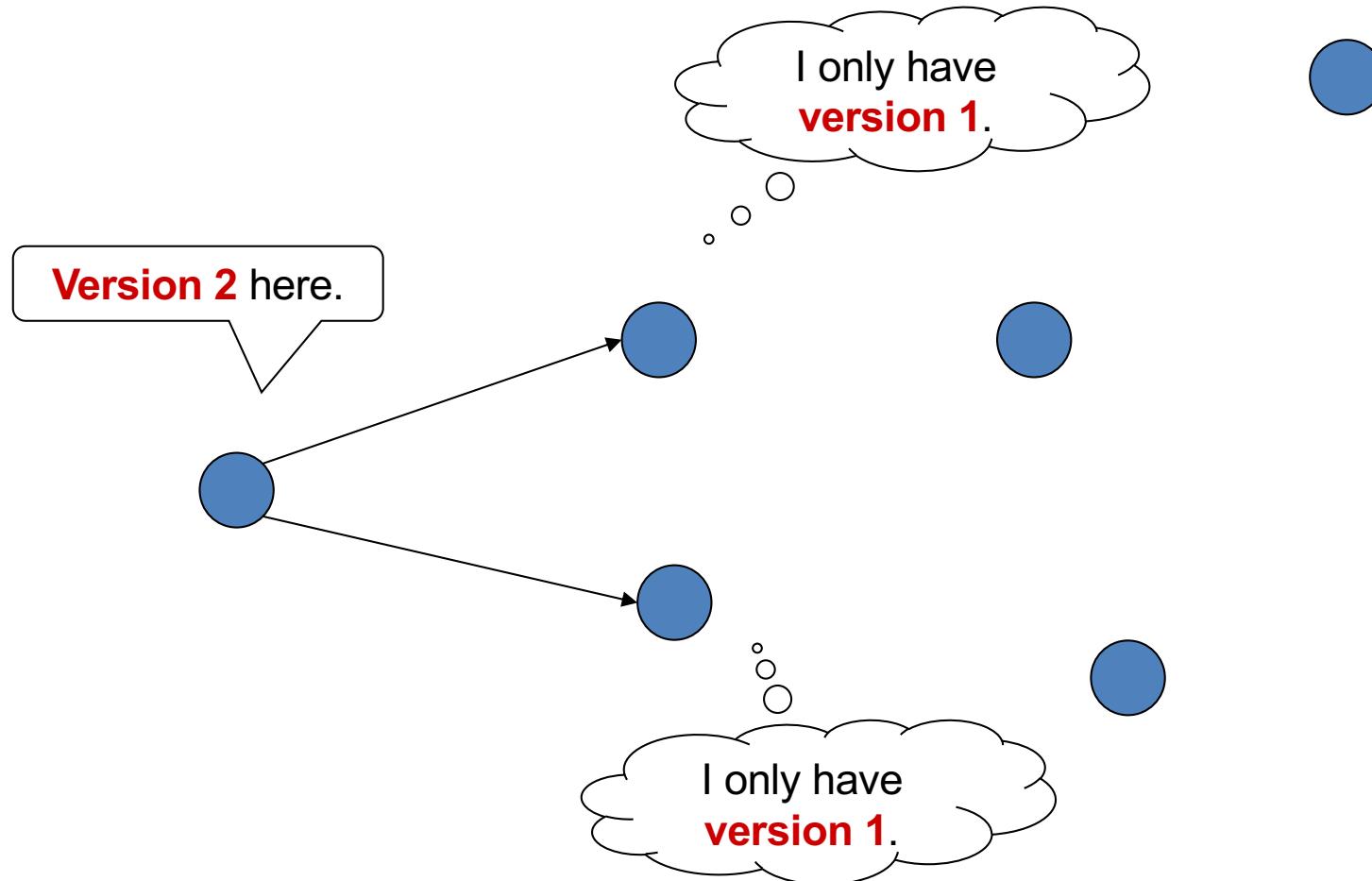
# Secure Code Image Programming in WSN



H. Tan, D. Ostry, J. Zic and S. Jha, "Secure Multi-hop Network Programming With Multiple One-way Key Chains", IEEE Transactions on Mobile Computing (TMC), Vol 10(1), pp 16-31, Jan 2011

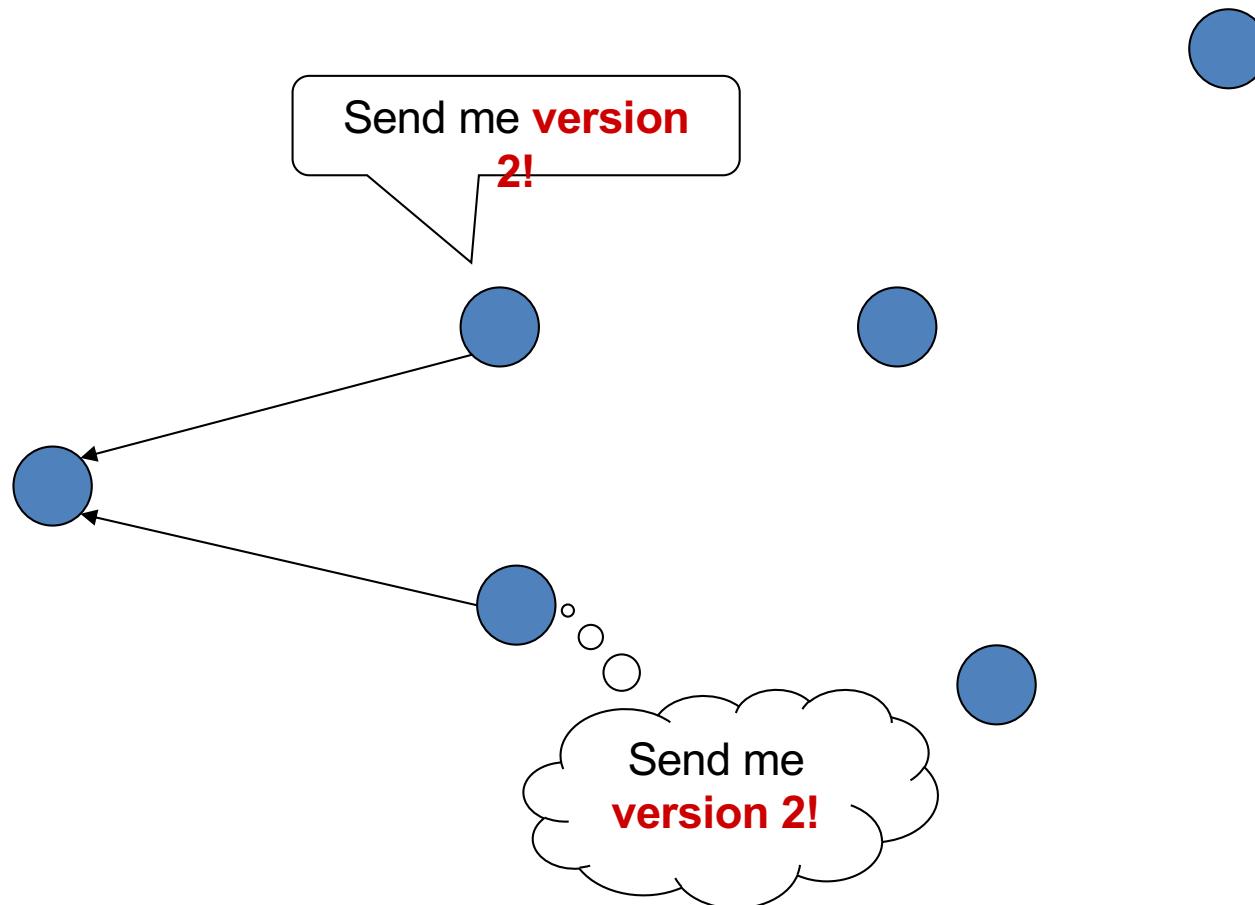
# Epidemic propagation (1)

## 1. Nodes periodically advertise



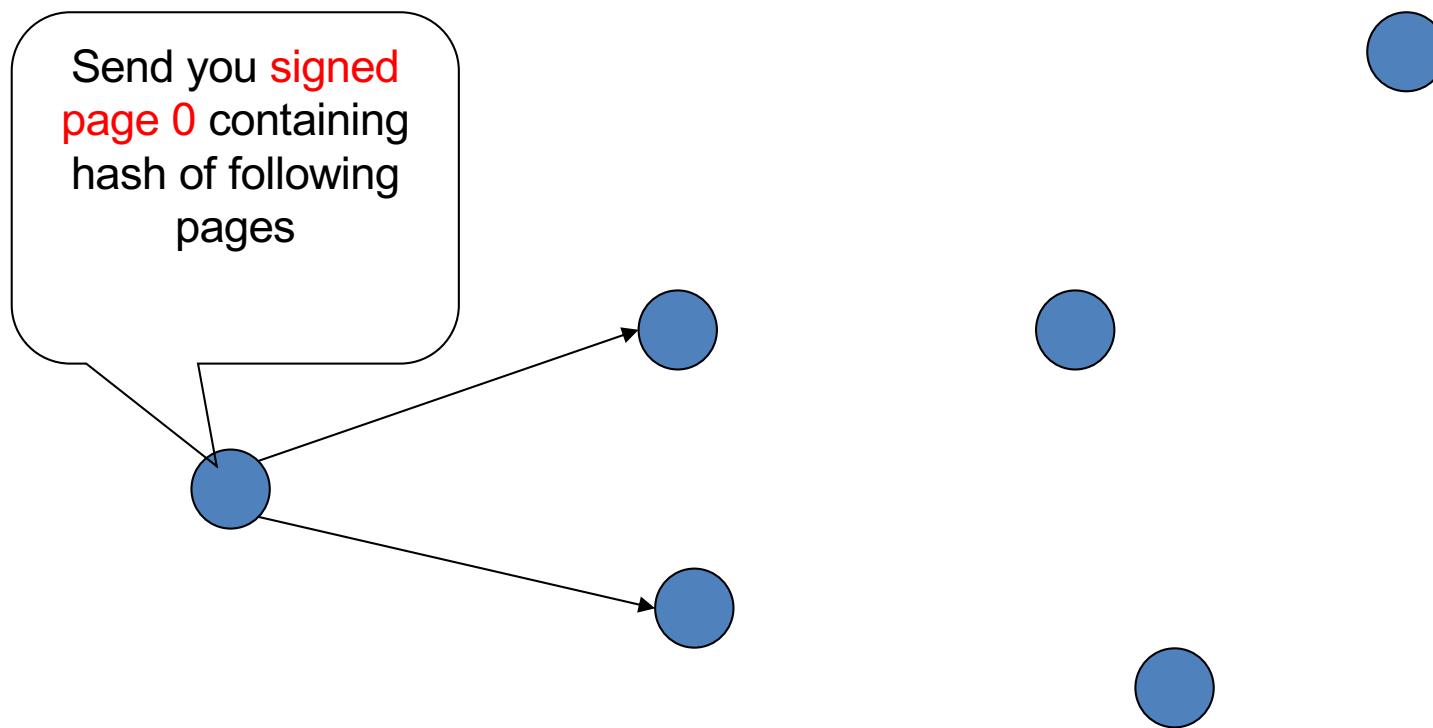
# Epidemic propagation (2)

## 2. Neighboring nodes request new version



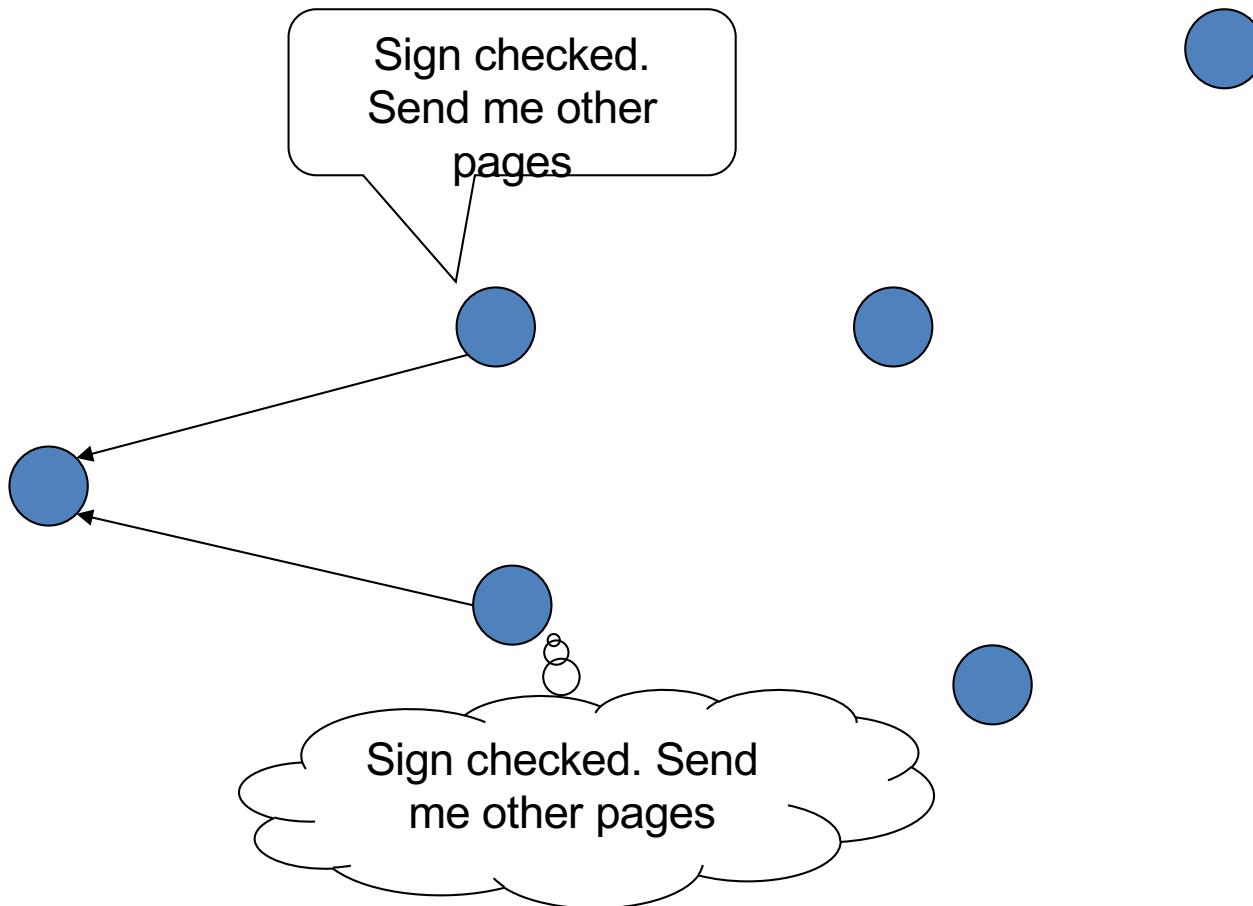
# Epidemic propagation(3)

3. Server sends signed hash of each page



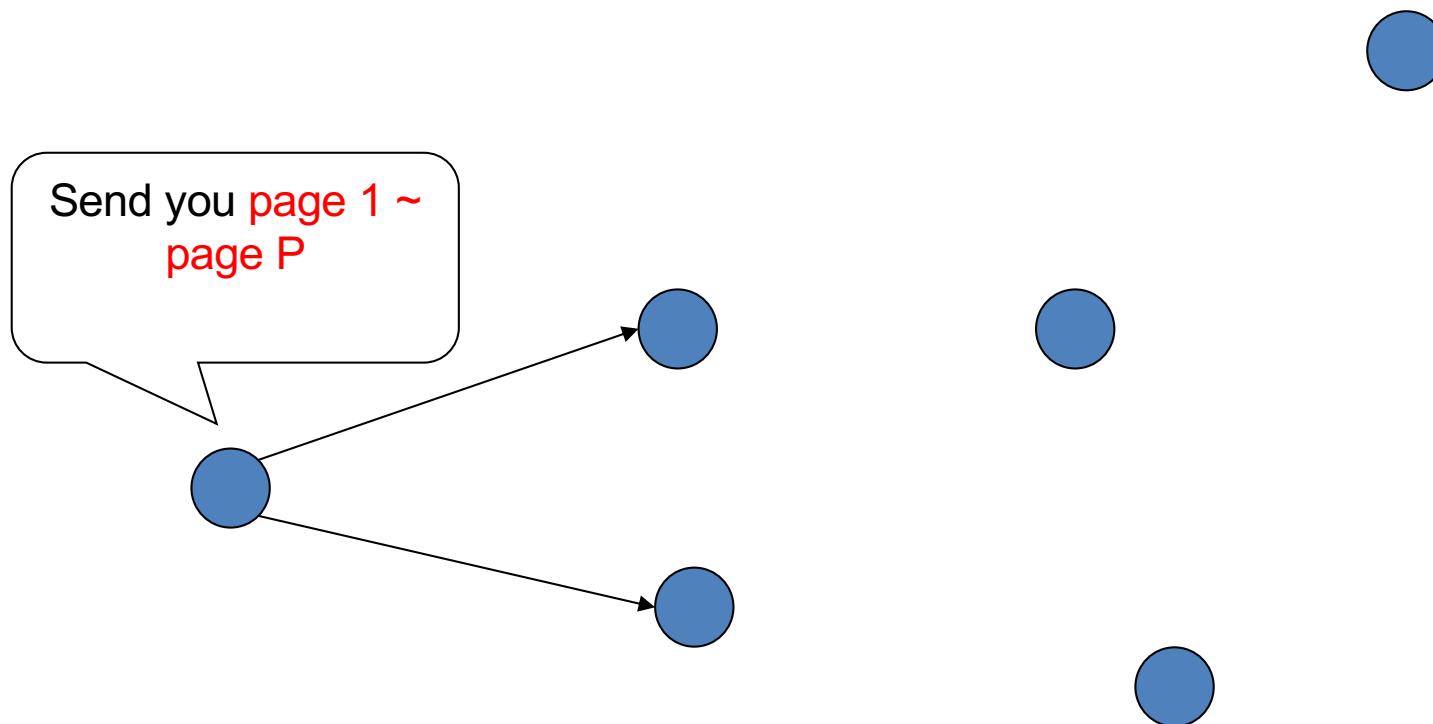
# Epidemic propagation (4)

## 4. Neighboring nodes request data



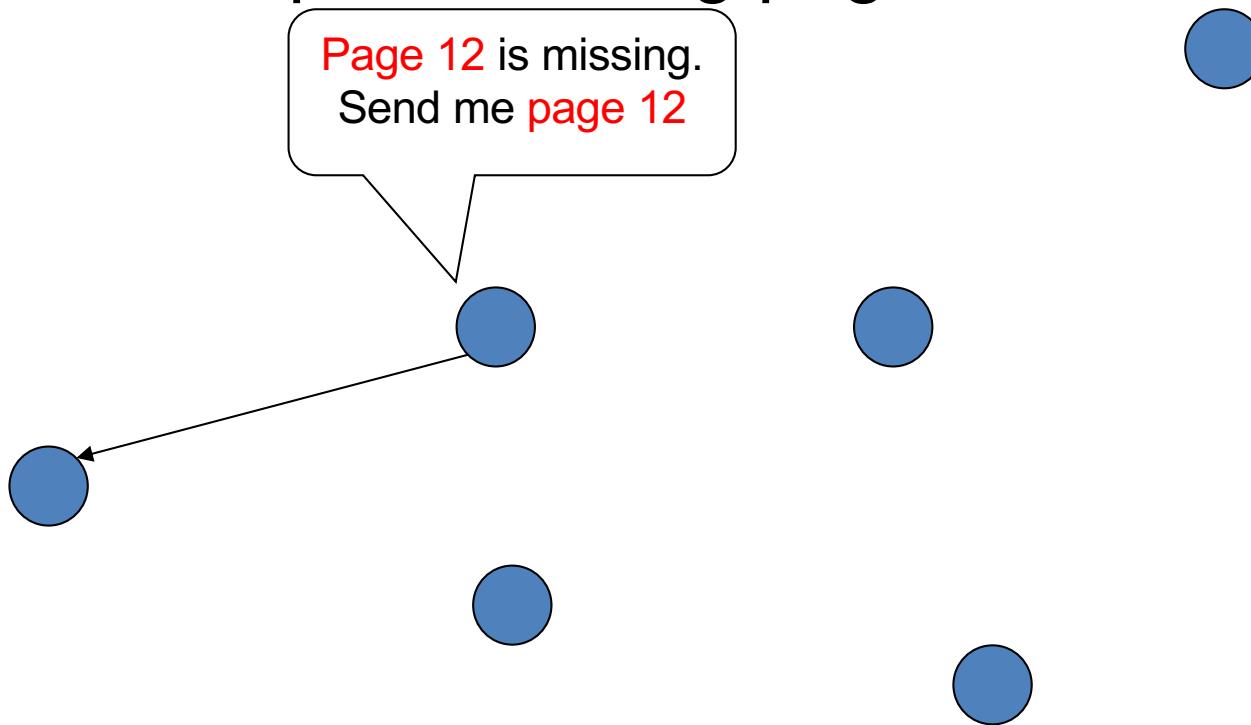
# Epidemic propagation (5)

## 5. Transmit each page



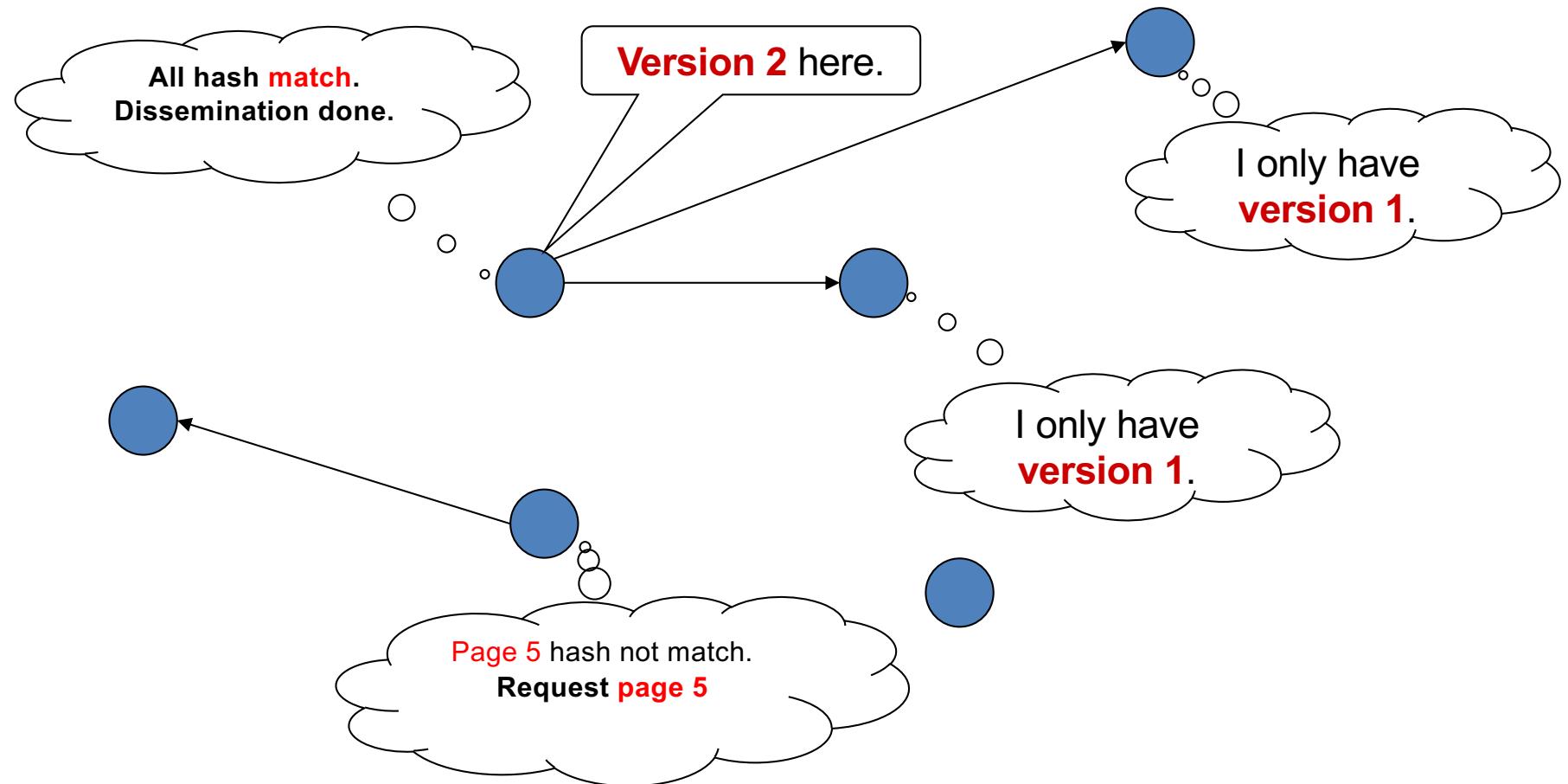
# Epidemic propagation (6)

## 6. Nodes request missing page



# Epidemic propagation (7)

## 7. Check hash of each page



# Threat Model

- Adversary has power laptop/computer
- External Attacks: adversary doesn't control any node
  - Eavesdrop, inject forged messages, replay intercepted messages, impersonate valid node, *Wormhole Attack* (fake non-existing links), *Sybil Attack* (one node presents several identities to defeat fault tolerance)
- Internal Attacks: take control of nodes
  - manipulate nodes until detected, intercept sensitive information even if encrypted, (selectively) drop packets, and launch Sybil attacks
- Adversary can distribute illegal code, drain battery, disconnect network etc.

-

安全指标

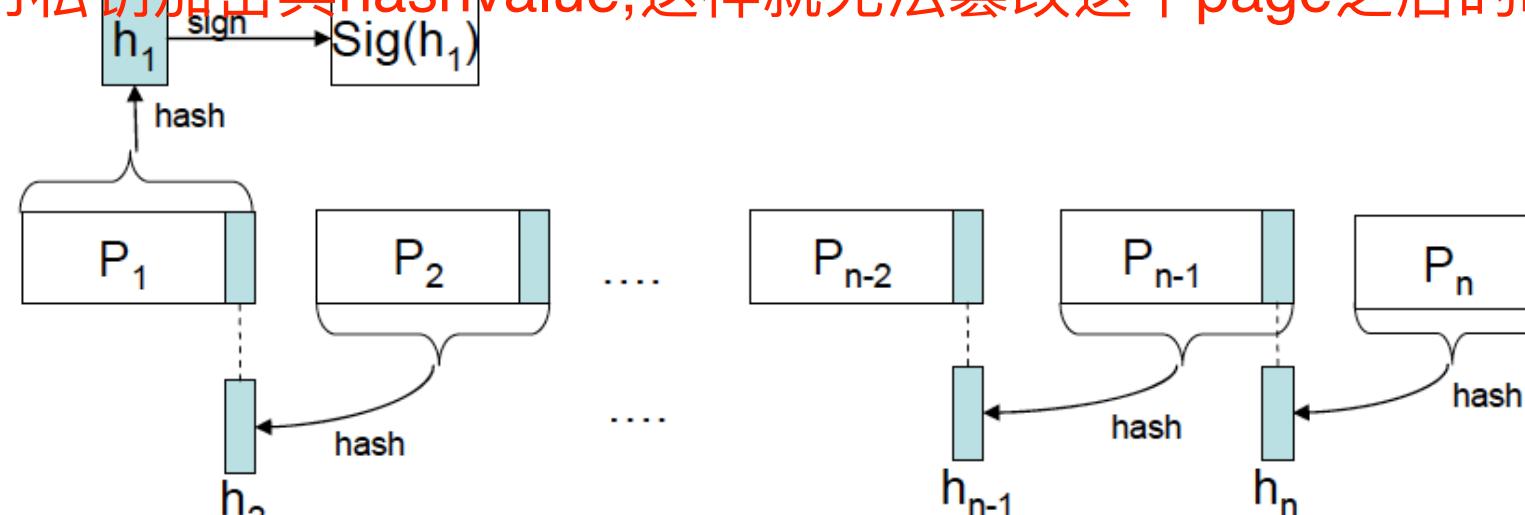
# Setting: Metrics

- Security metrics
  - Can external adversary forge a message?
  - Can single or several receivers forge a message that at least one other receiver accepts?
- Efficiency metrics      效率指标
  - Communication overhead
  - Computation overhead
  - Storage overhead
  - Delay for authentication / signature
  - Resilience to packet loss

# Code Dissemination – Hash Chain

- Image fragmented into fixed size segments – called pages
- Calculate hash value ( $h_n$ ) of last page  $P_n$  and append this to previous page  $P_{n-1}$ , until first page is reached
- Hash value of first page  $P_1$  signed with private key of base station (BS)
- Receivers verify this using public key of BS and recursively authenticate each page

把后一页的hash append到他前一个,以此类推,直到第一个page,第一个page需要用私钥加密其hashvalue,这样就无法篡改这个page之后的hashvalue

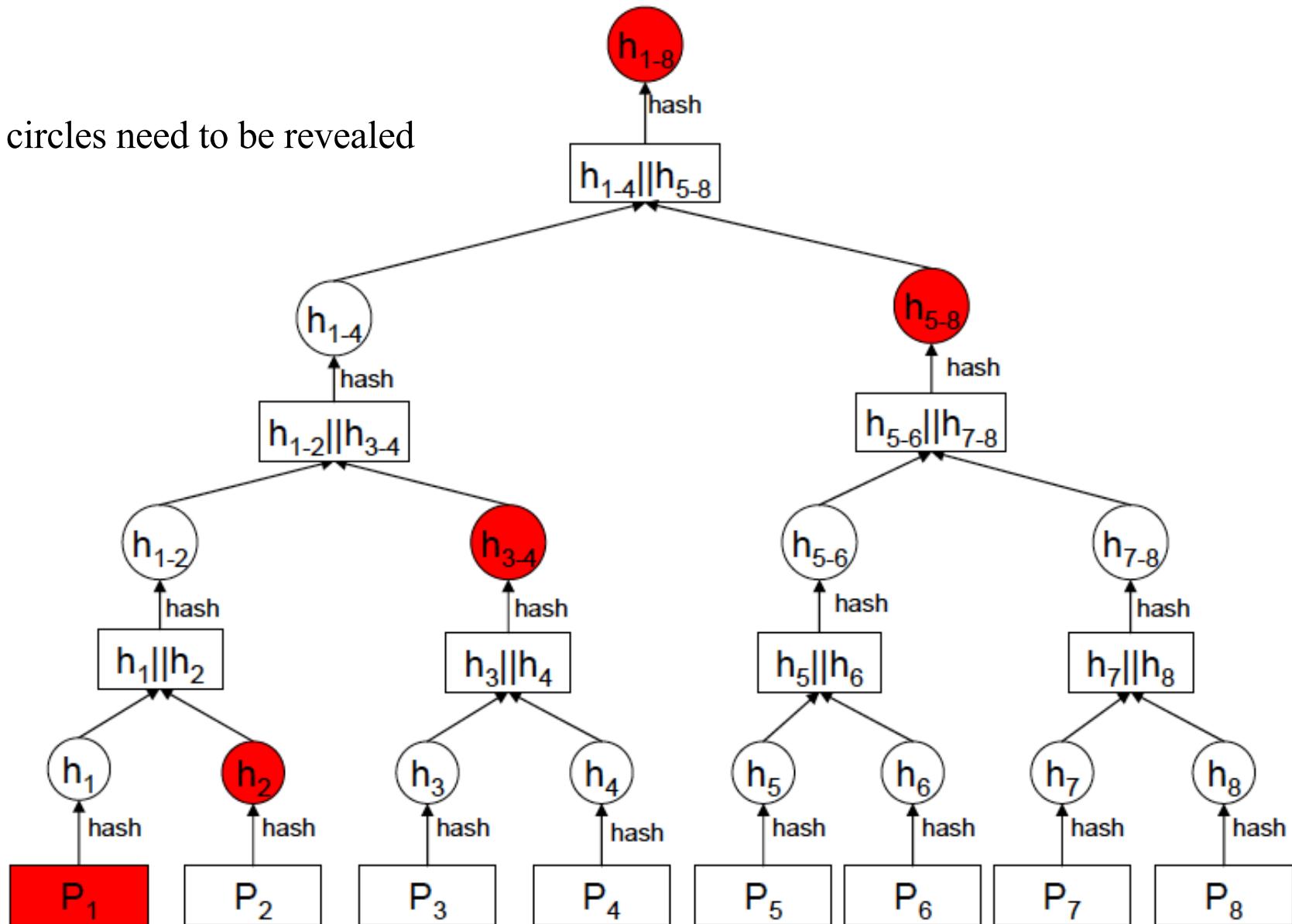


# Merkle Hash Tree

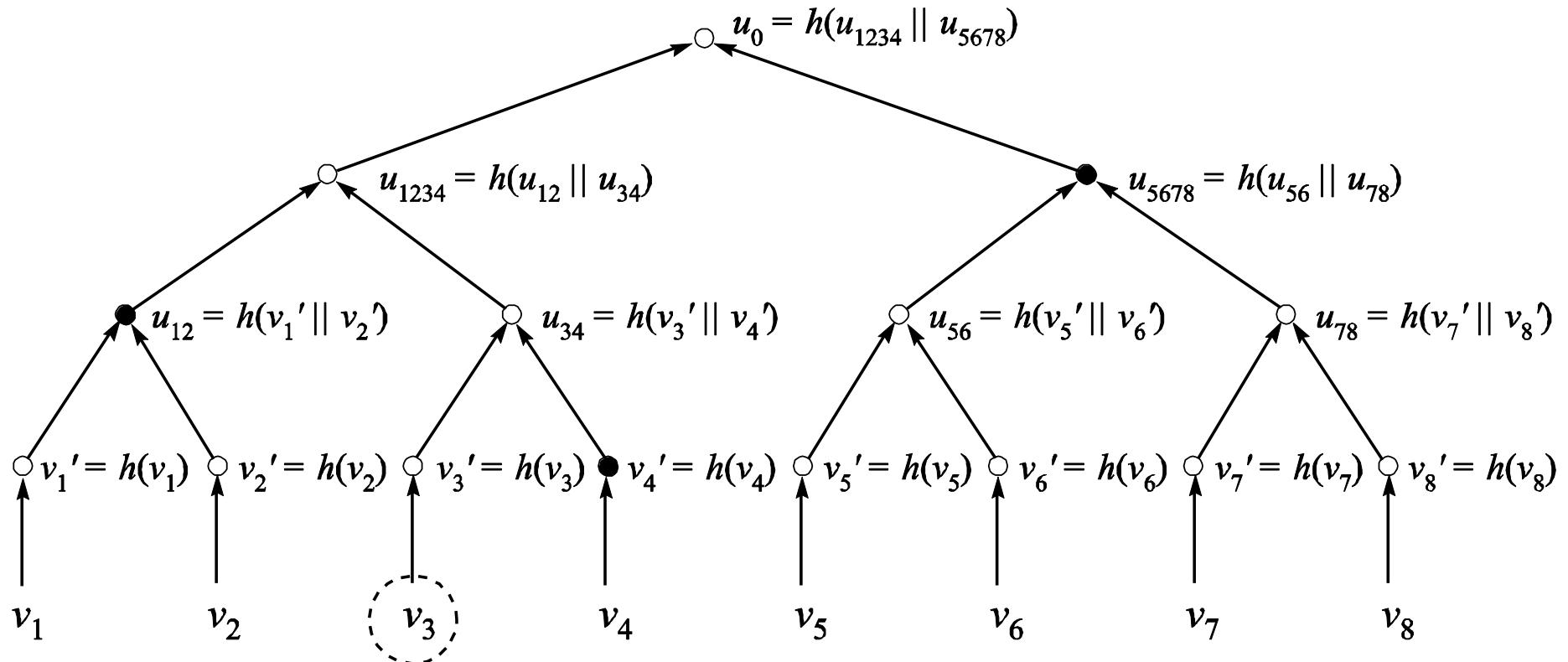
- Hash Chain elements can only be revealed sequentially
  - May not be acceptable for many applications e.g. P2P
- Merkle-trees : allowing for the pre-authentication of a set of values with a single digital signature (on the root  $u_0$  of the tree) and for the revelation of those values in *any* order
- when revealing a value  $v_i$ , reveal all the values assigned to the sibling vertices on the path from  $v_i'$  to the root
  - One way hash property ensures: this disclosure not sufficient to calculate any other unrevealed  $v_j$

# Code Dissemination-MT Example

\*Red circles need to be revealed



# Merkle Tree Another Example



- Authenticate  $v_3, v_3$  is revealed together with  $v_4', u_{12}, u_{5678}$
- verifiers hash the revealed values appropriately and check if the result is  $u_0$ 
  - $u_0 = H(H(u_{12} \parallel H(H(v_3) \parallel v_4')) \parallel u_{5678})$

-

# Code Dissemination - MT

- We could further divide each page into packets
- Take hash of each packet
- Construct a Merkle tree of these hash values and get root for each page
- Recursive hash value of the roots of each page is done using Hash-Chain
- Digital signature for the first root value is sent
- Along with root value, also need sibling vertices as discussed earlier

# Signature Based Attack

- Signature expensive operation on small embedded devices
- Attacker keeps sending forged data
- Node depletes energy in doing signature verification
- These approaches need to know number of data-packets in advance
  - may not be available in many applications

-

# Small RSA Signature

- Idea
  - Use short-lived small RSA keys (e.g., 384 bit)
  - Periodically send out new public key signed with strong signature
    - 48 byte signature per packet
    - Signature generation ~0.1ms, verification ~10us
- Advantages
  - Relatively low computation overhead
  - No buffering, no verification delay
  - Scalable
- Disadvantages
  - Relatively high communication overhead (> 50 bytes/packet)
  - Need time synchronization
  - Not perfectly robust to packet loss

# Learning Outcomes

- Appreciate how Broadcast/multicast fundamentally changes protocol design space for authentication.
- Understand tradeoff between reliability and security
  - Key exchange etc must be reliable?
- Understand hash chain and Merkle tree algorithms and their application for security.
- Understand various threats in adhoc, wireless sensor networks and IoT
- Appreciate that different points of the design space have different “best solution”

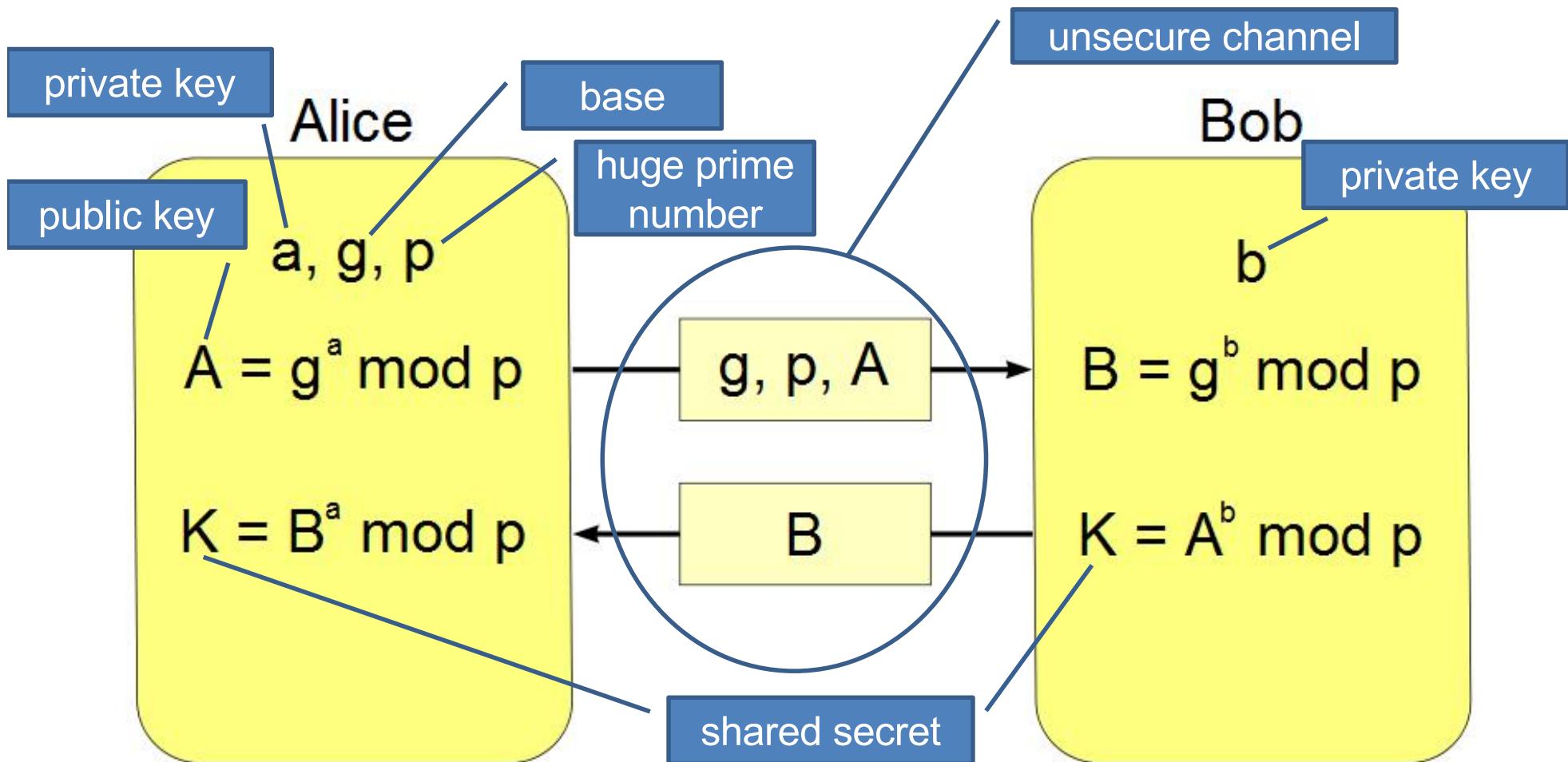
# Reference List

- MERKLE TREE
  - *Appendix-A, L. Buttyan and J. P. Hubaux*, Security and Cooperation in Wireless Networks
  - A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001)*, pages 35–46. Internet Society, February 2001
- Seluge/Deluge
  - Sangwon Hyun, Peng Ning, An Liu, and Wenliang Du. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. In IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks, pages 445–456, 2008.
- A good paper on Broadcast Encryption scheme
  - D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005*, pages 258–275, 2005.

# Reference (Contd)

- NaCL (Salt) Network and Cryptography Library (<http://nacl.cr.yp.to>)
- [\*\*RFC 4347 - Datagram Transport Layer Security\*\*](#)
- For interactive ECC curve etc
  - [https://www.certicom.com/ecc tutorial/ecc javaCurve.html](https://www.certicom.com/ecc_tutorial/ecc_javaCurve.html)
  - ECC Video (<https://www.youtube.com/watch?v=l6jTFxQaUJA>)
- ACKNOWLEDGMENT: Foils contributed by Phd students Hailun Tan, Jun Kim Young
  - Some adapted from Prof Adrian Perrig.
  - ECC foils are modified from Michael Karls's version.
- Optional Read:
  - [Shamir, Adi \(1979\), "How to share a secret", \*Communications of the ACM\*, \*\*22\*\* \(11\): 612–613.,](#)
  - R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, pages 708–716, March 1999.

# Diffie-Hellman key exchange



$$K = A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = g^{ab} \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = B^a \text{ mod } p$$

Alice's private key = 5, Bob's private key = 4,  $g=3$ ,  $p=7$

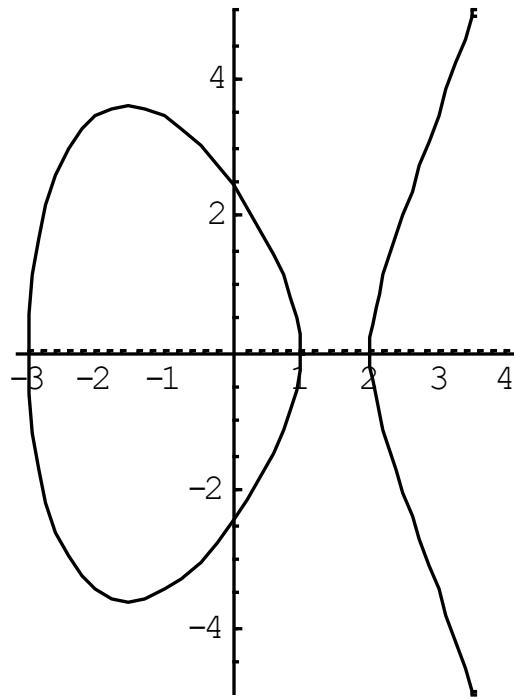
Alice's public key =  $3^5 \text{ mod } 7 = 5$ , Bob's public key =  $3^4 \text{ mod } 7 = 4$

Alice's shared key =  $4^5 \text{ mod } 7 = 2$ , Bob's shared key =  $5^4 \text{ mod } 7 = 2$

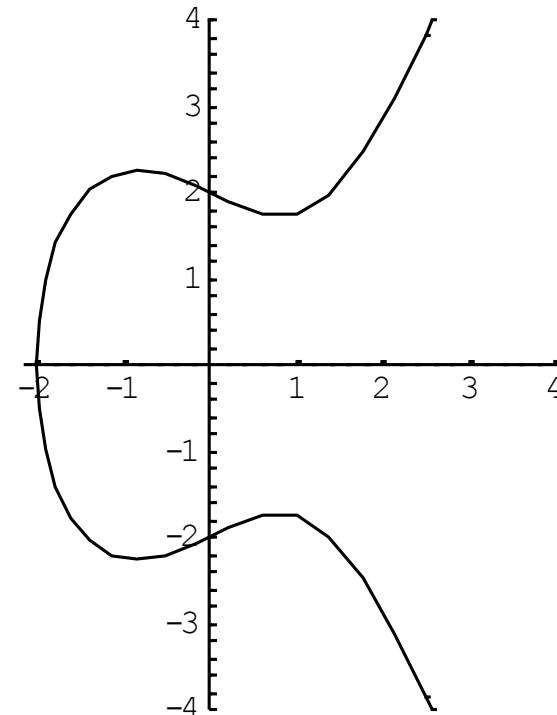


# Examples of Elliptic Curves (Optional)

- $y^2 = x^3 - 7x + 6$

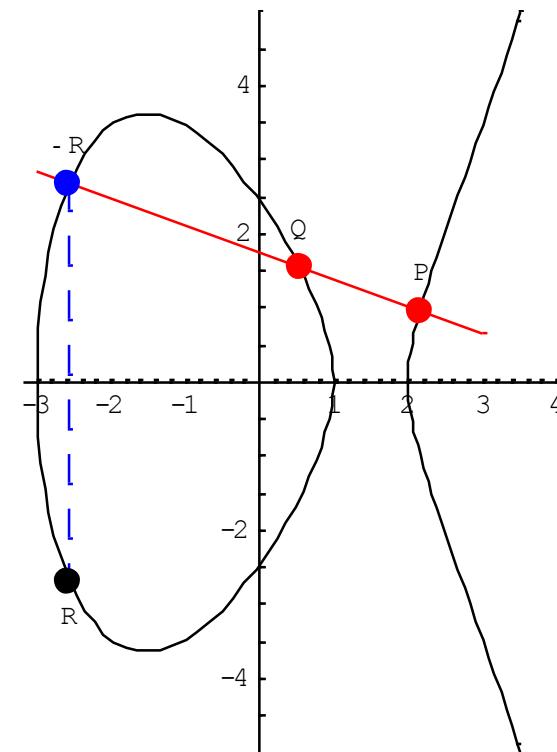


- $y^2 = x^3 - 2x + 4$



# Adding Two Points (Geometrically): $x_P \neq x_Q$

- We skip maths/algebraic details (beyond scope)
  - The line L through P and Q will intersect the curve at one other point.
  - Call this third point -R.
  - Reflect the point -R about the x-axis to point R.
  - $P+Q = R$
- $y^2 = x^3 - 7x + 6$



# Point Doubling: $x_P=x_Q$ and $y_P = y_Q$

- Since  $P = Q$ , the line  $L$  through  $P$  and  $Q$  is tangent to the curve at  $P$ .
  - Again  $L$  will intersect the curve at another point,  $-R$ .
  - As in Case 1, reflect  $-R$  about the  $x$ -axis to point  $R$ .
  - $P+P = R$
  - Notation:  $2P = P+P$
  - *Basically this computation (and variants) is more efficient than the standard Diffie-Hellman*
  - *Crypto: Let  $P$  and  $Q$  be two points on an elliptic curve such that  $kP = Q$ , where  $k$  is a scalar. Given  $P$  and  $Q$ , it is hard to compute  $k$ .*
- $y^2 = x^3 - 7x + 6$

