PASTA: PASsword-based Threshold Authentication

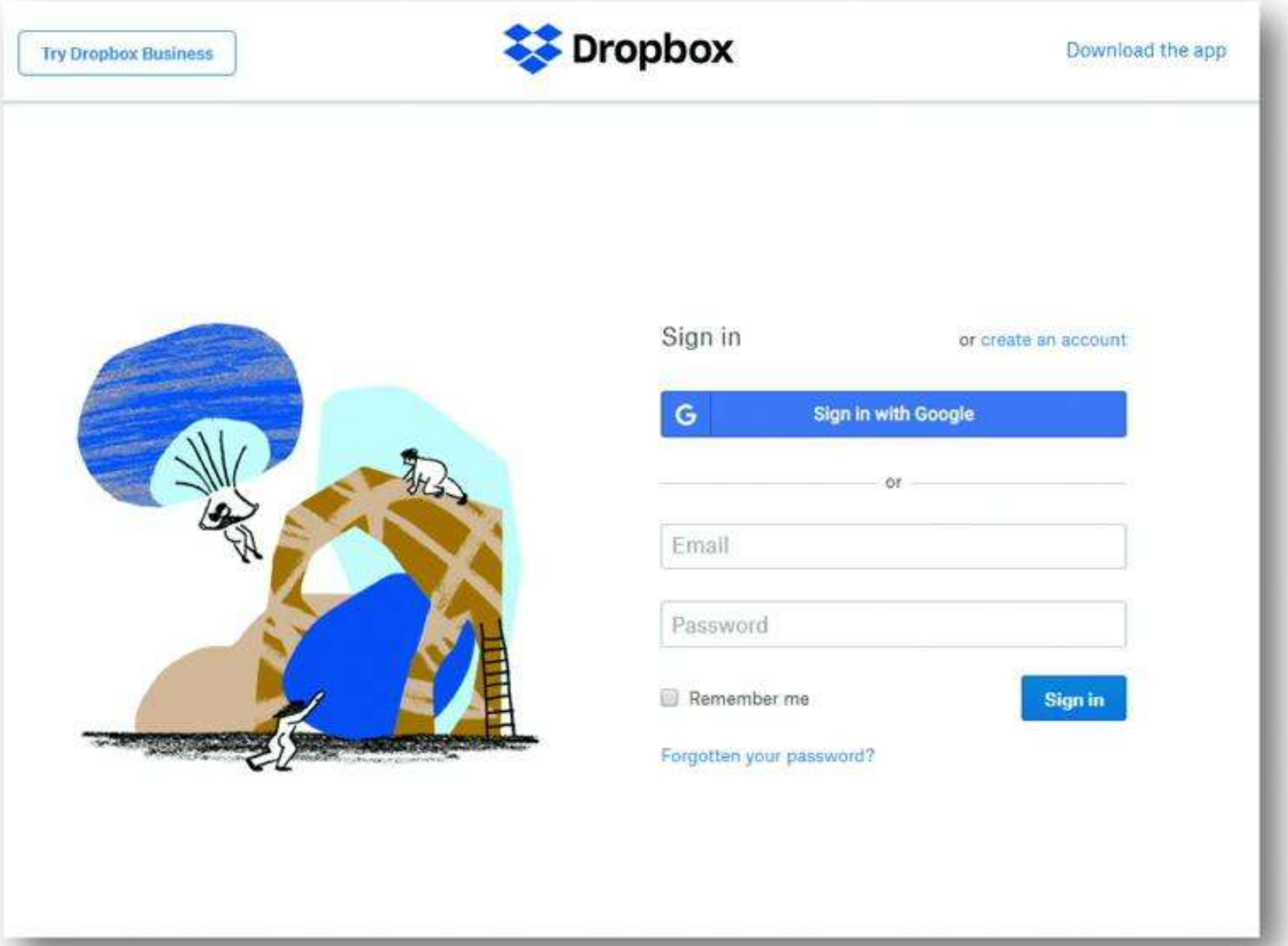
Peihan Miao

June 24, 2019

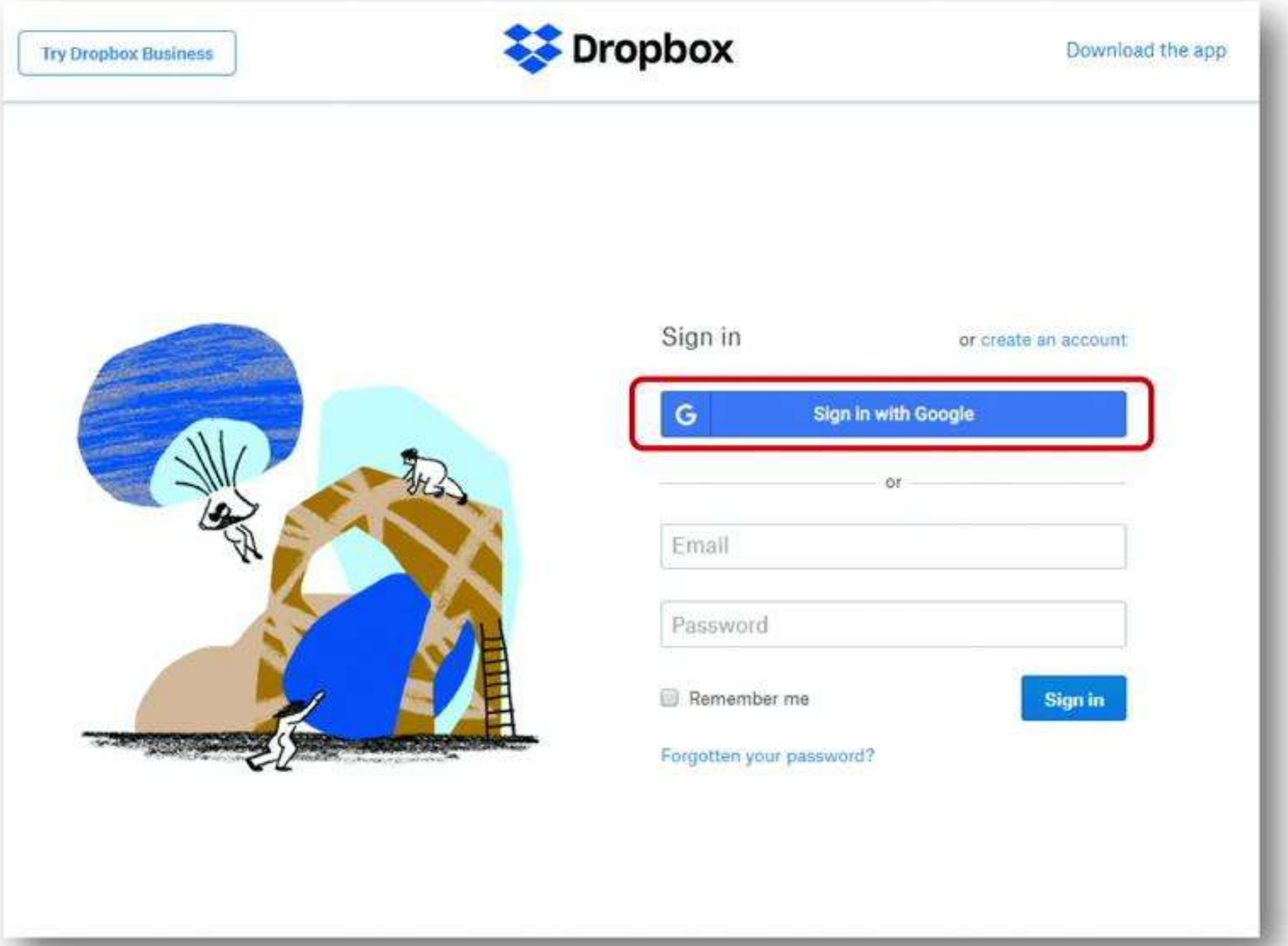
Joint work wih Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee.

What problem is **PASTA** trying to solve?

OAuth


Try Dropbox Business
Dropbox
Download the app
Sign in
or create an account
Sign in with Google
or
Email
Password
Remember me
Sign in
Forgotten your password?

OAuth


Try Dropbox Business
Dropbox
Download the app
Sign in
or create an account
Sign in with Google
or
Email
Password
Remember me
Sign in
Forgotten your password?

OAuth

Google

OAuth

Google

OAuth

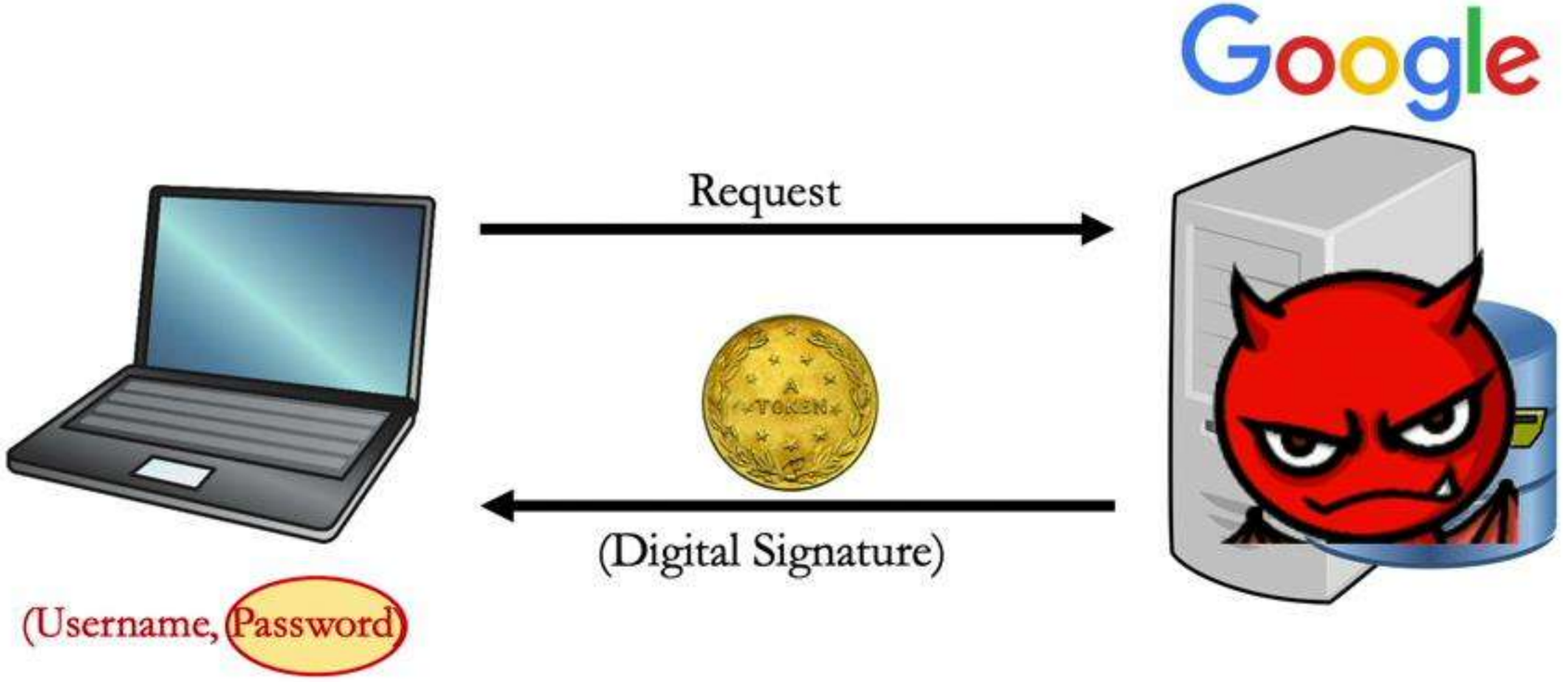
(Username, Password)

OAuth

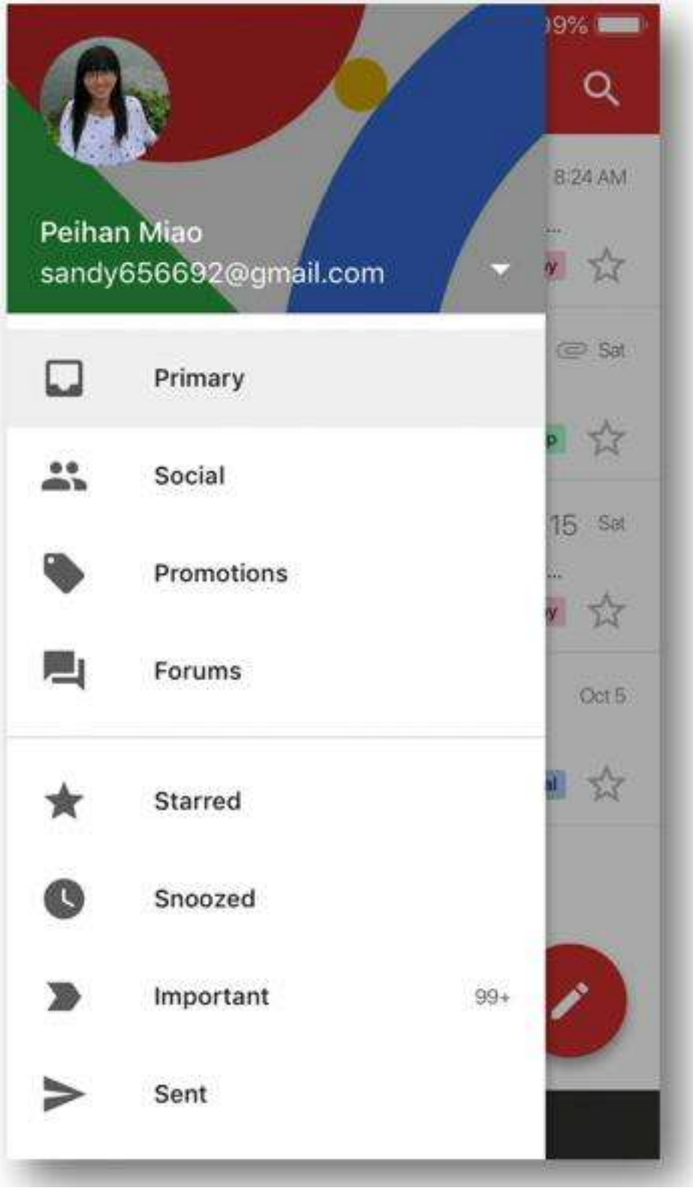
Request

(Username, Password)

OAuth



JSON Web Token (JWT)



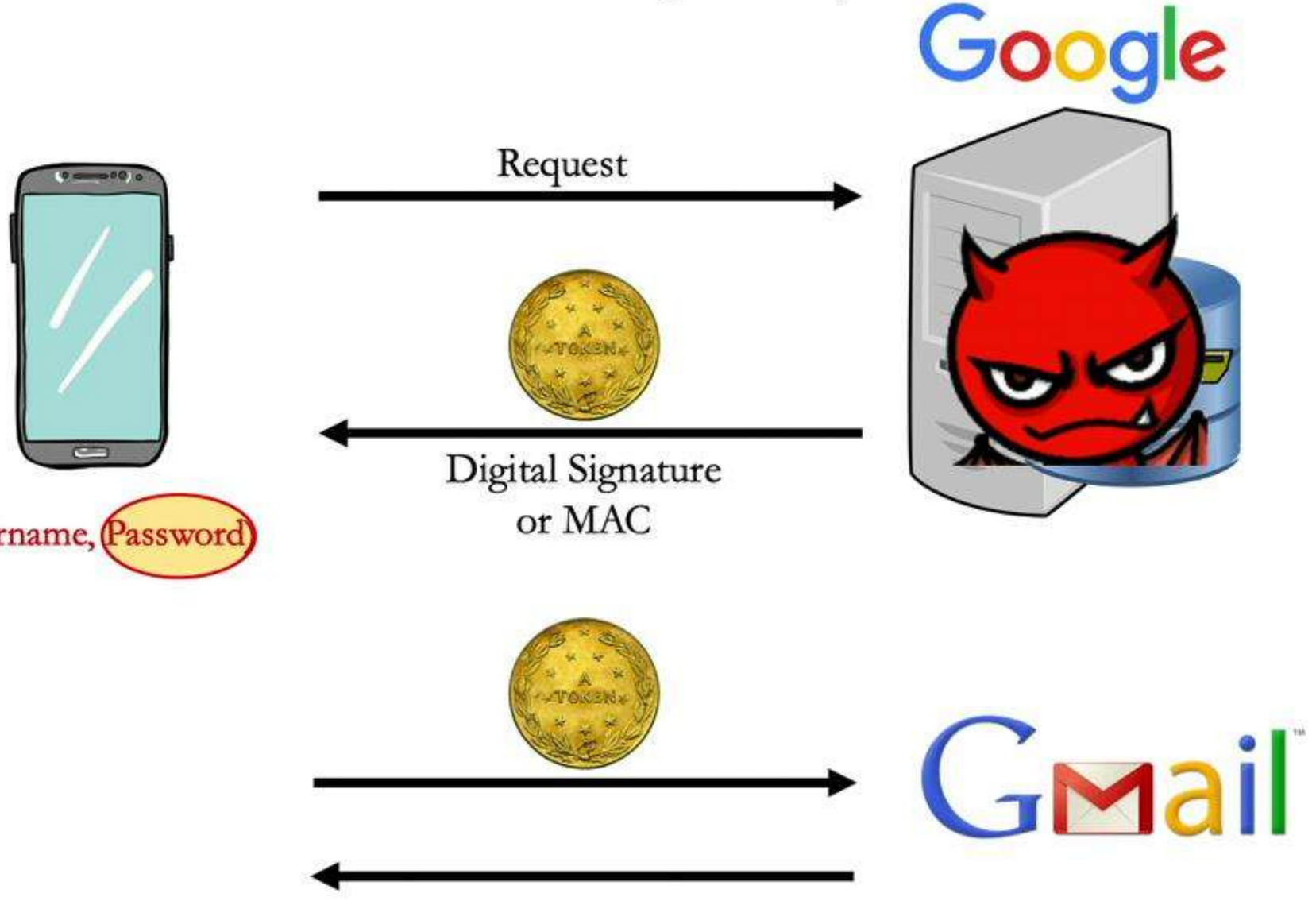
JSON Web Token (JWT)

(Username, Password)

Request

Digital Signature
or MAC

JSON Web Token (JWT)

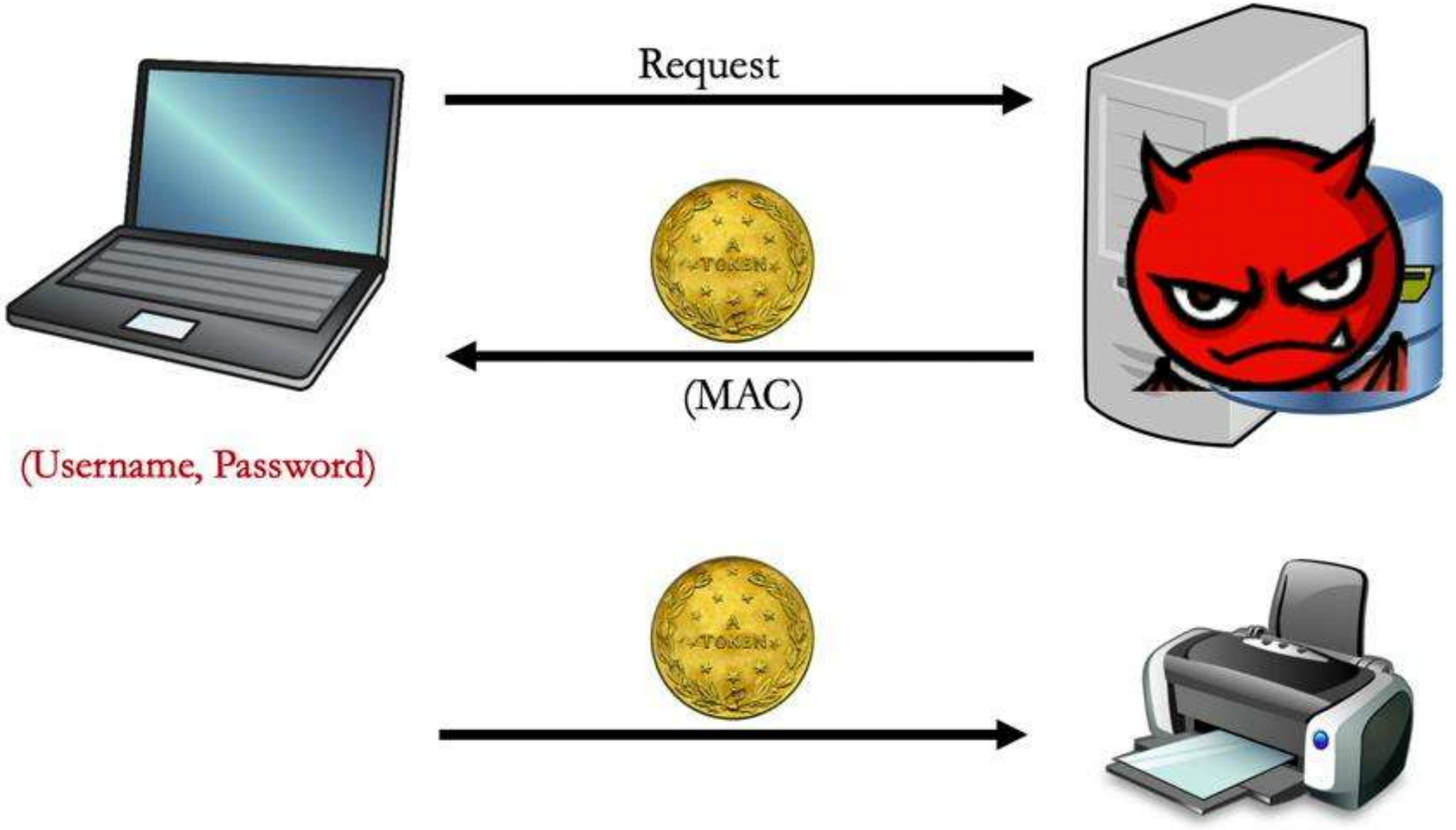


Kerberos

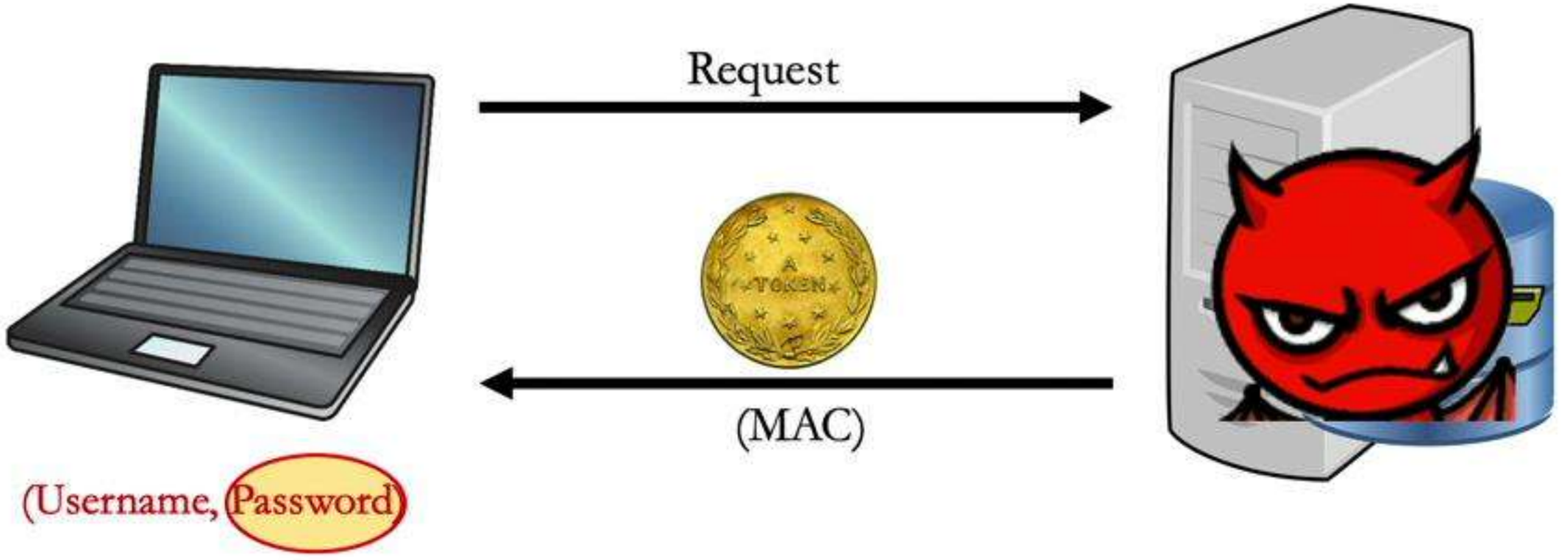
Kerberos

Kerberos

Kerberos



Kerberos



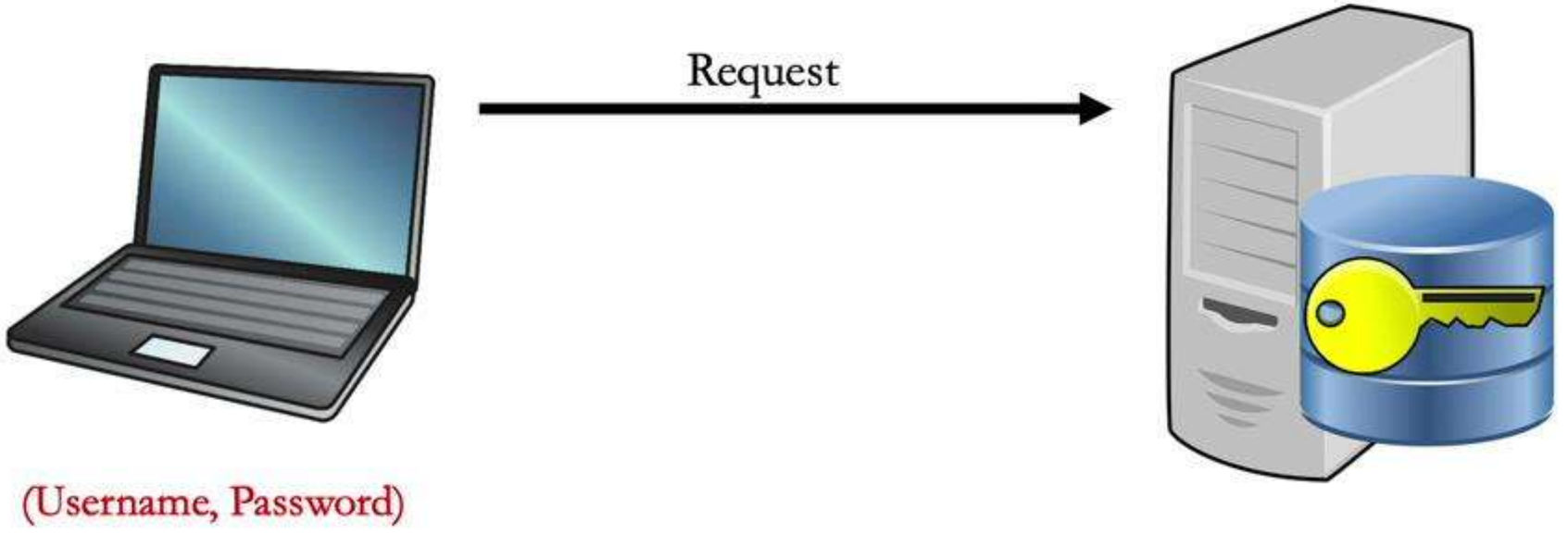
Password-Based Token Generation

Password-Based Token Generation

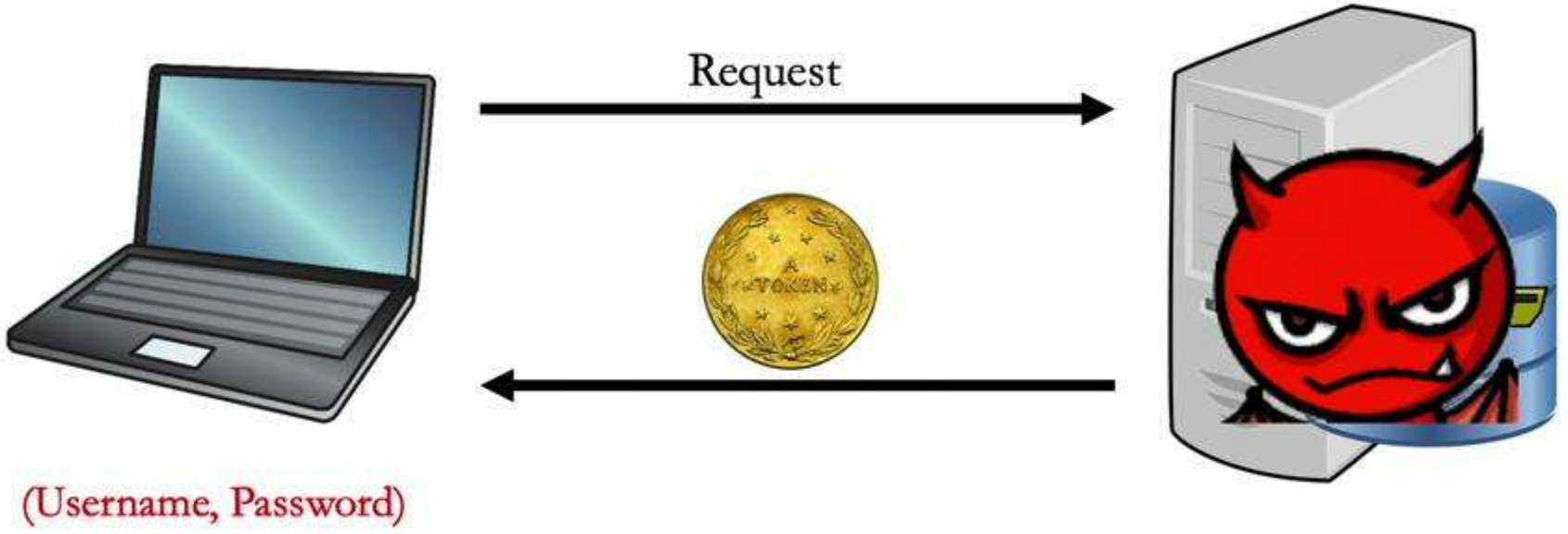
Password-Based Token Generation

(Username, Password)

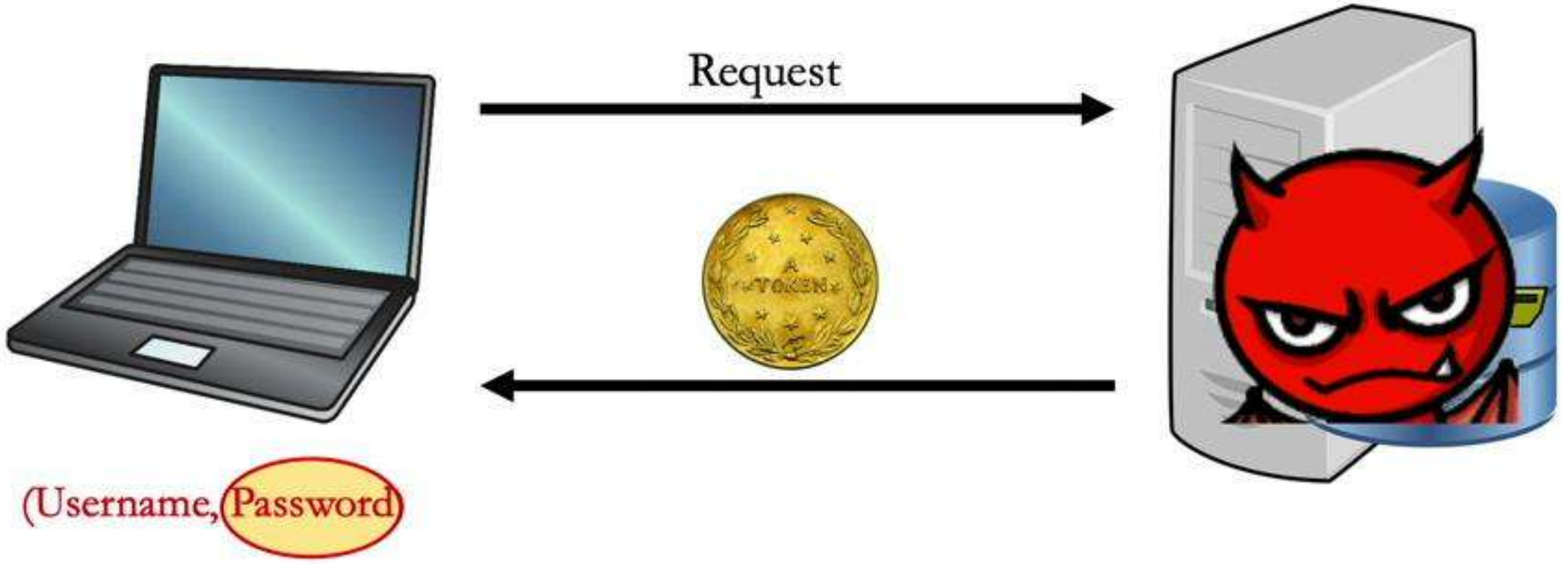
Password-Based Token Generation



Password-Based Token Generation



Password-Based Token Generation

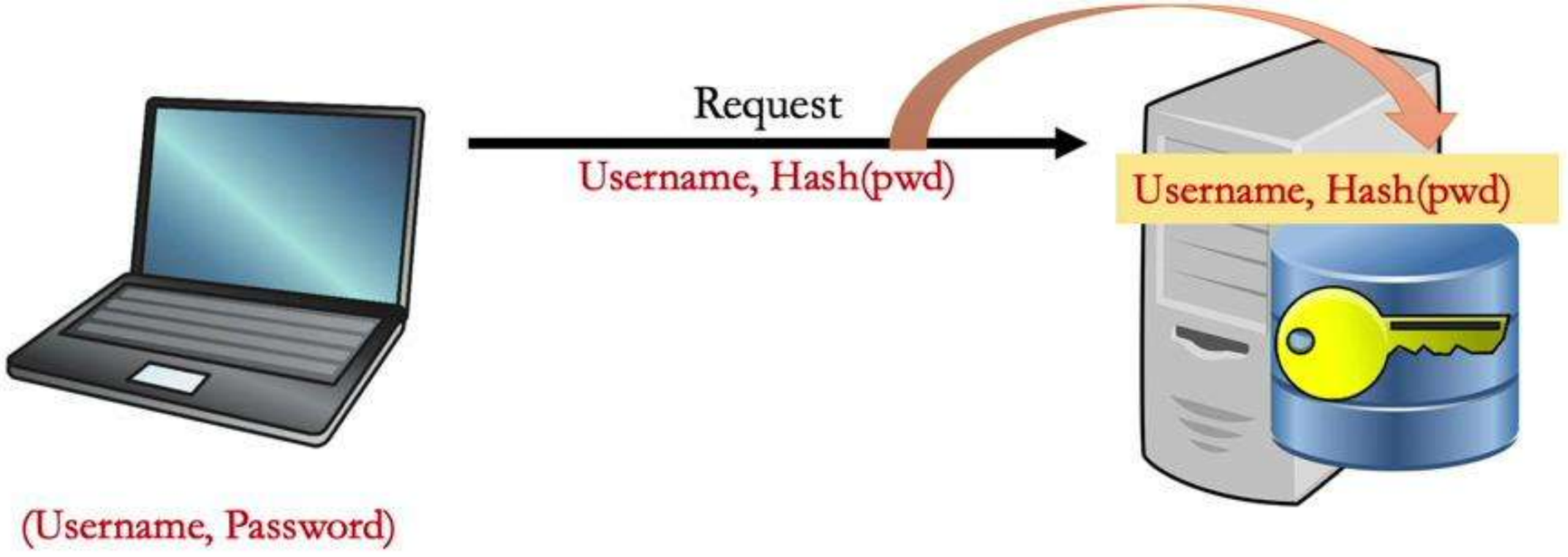


Password Safety

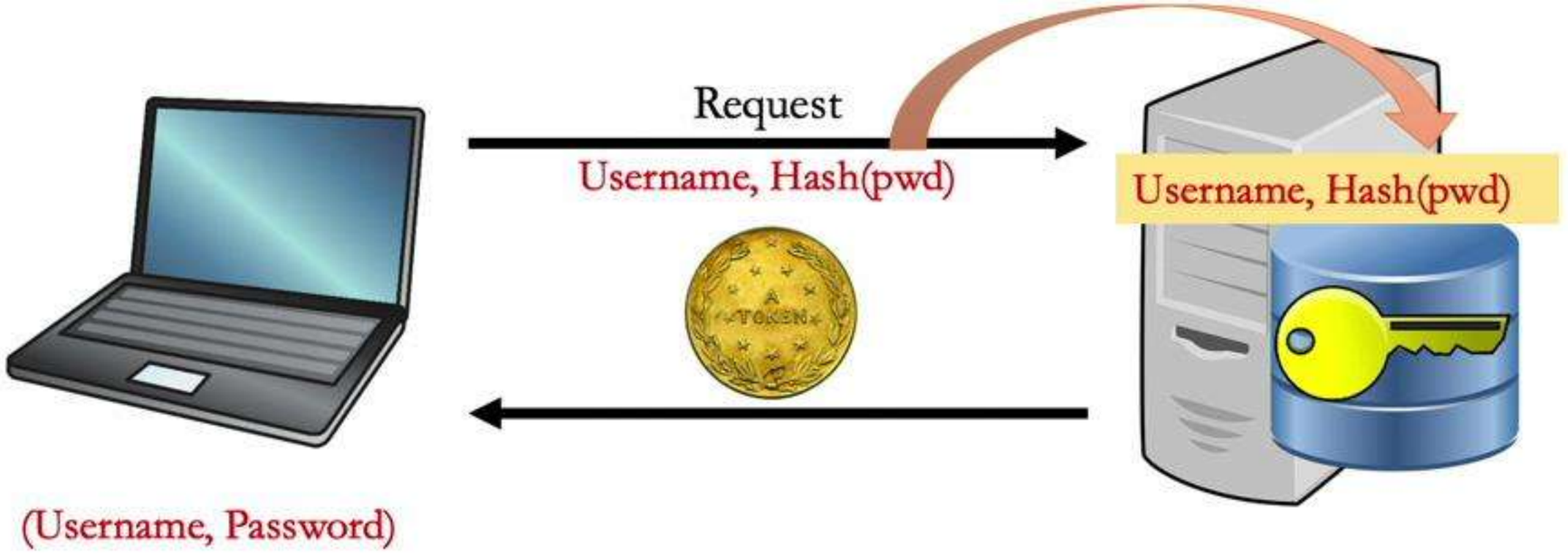
(Username, Password)

Password Safety

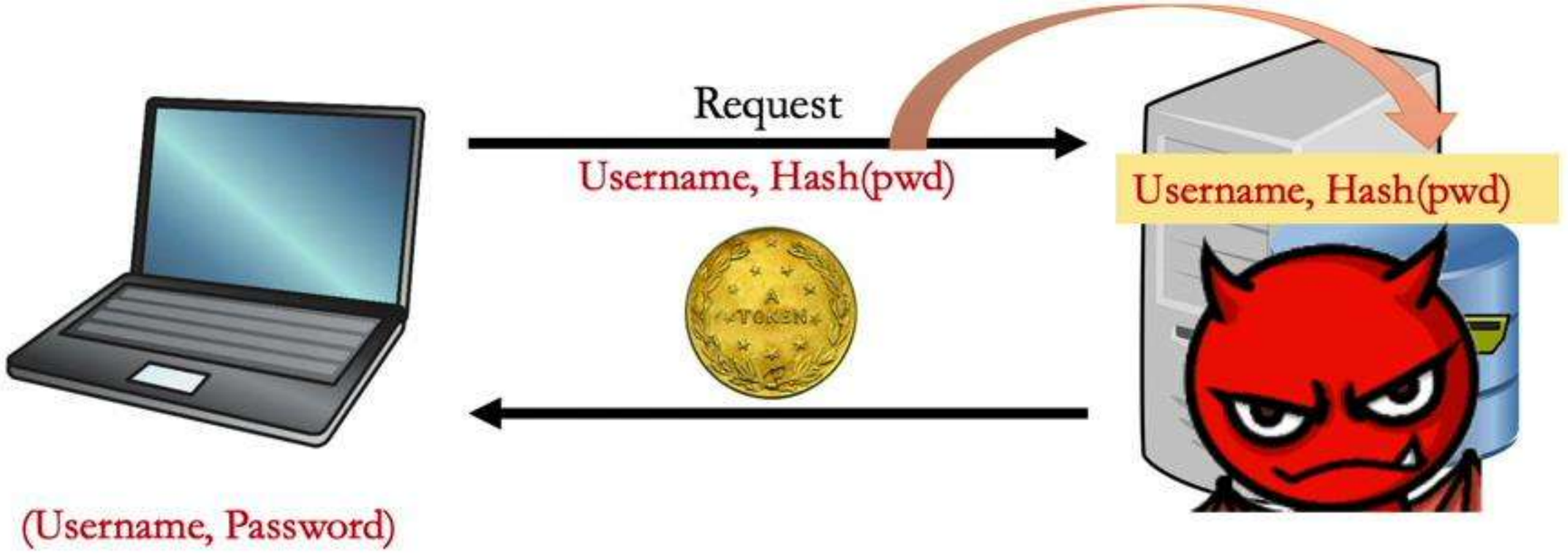
Password Safety



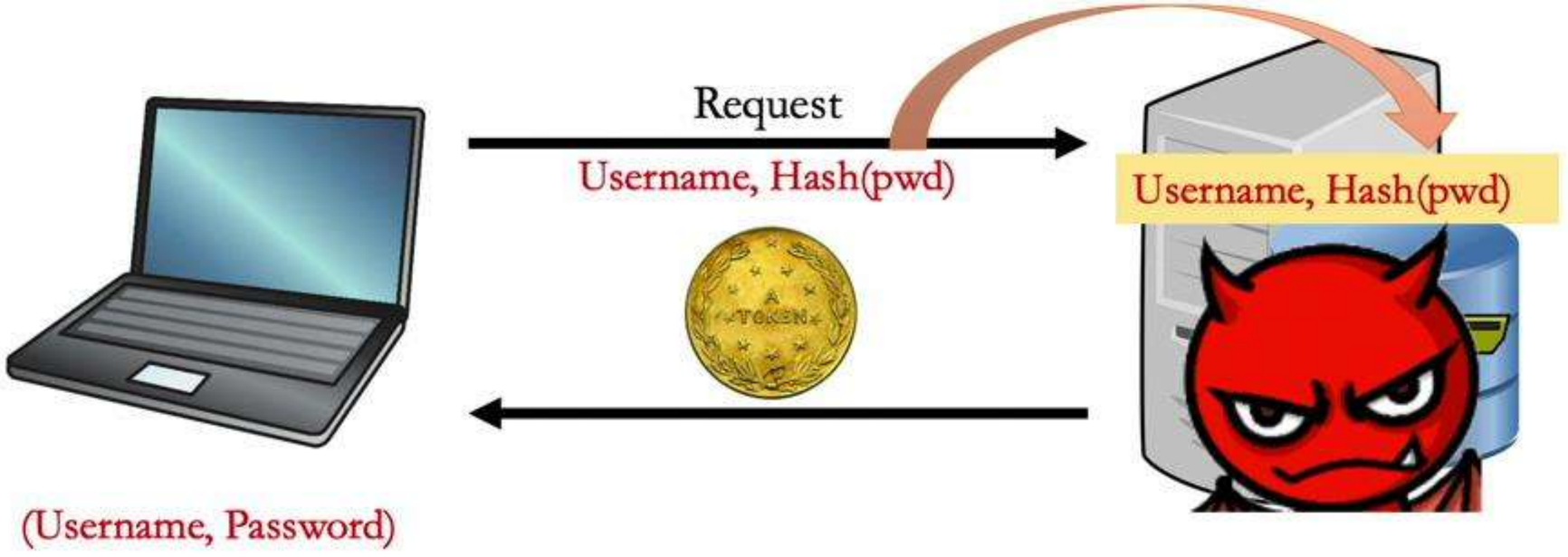
Password Safety



Password Safety

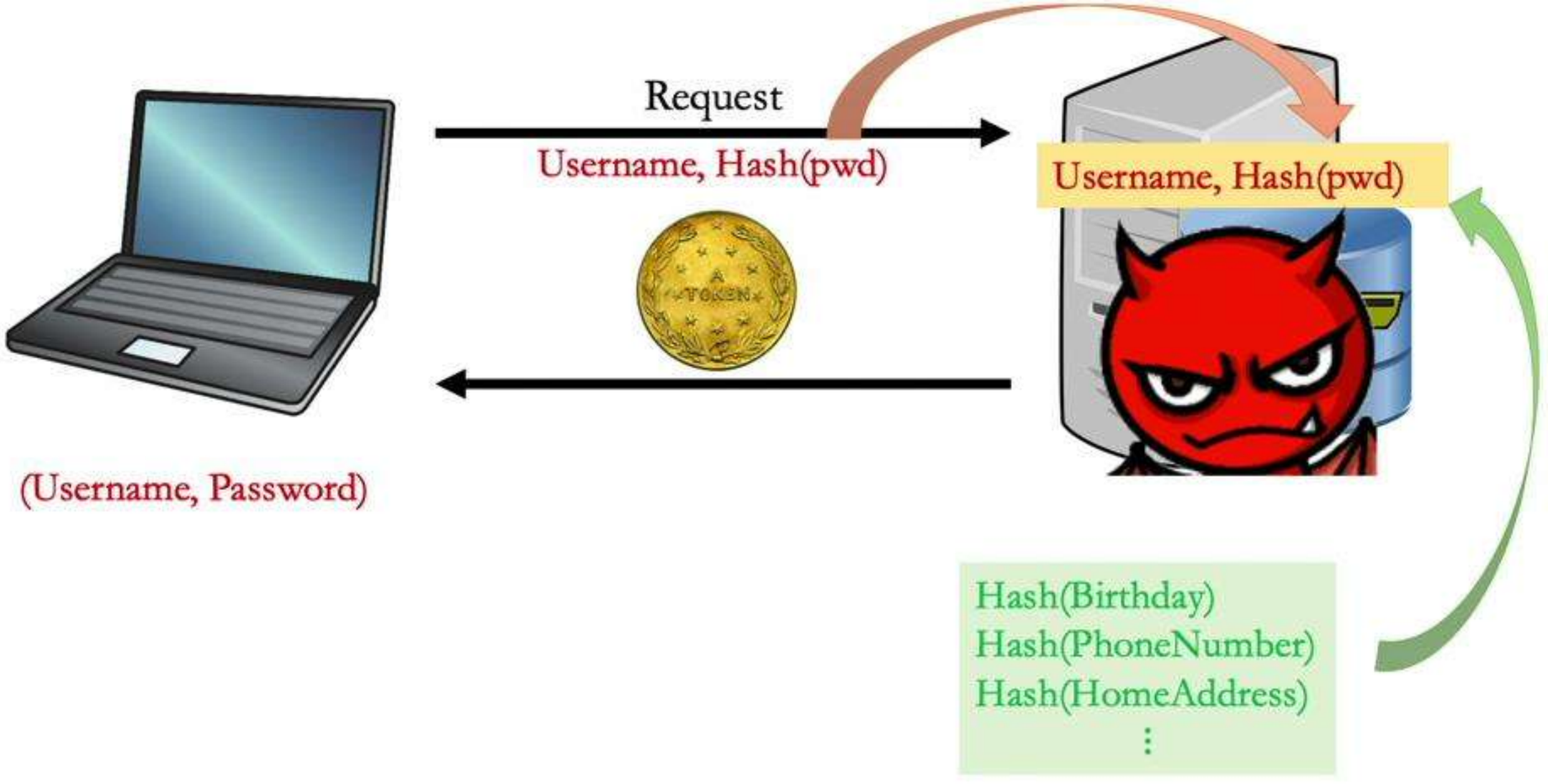


Password Safety

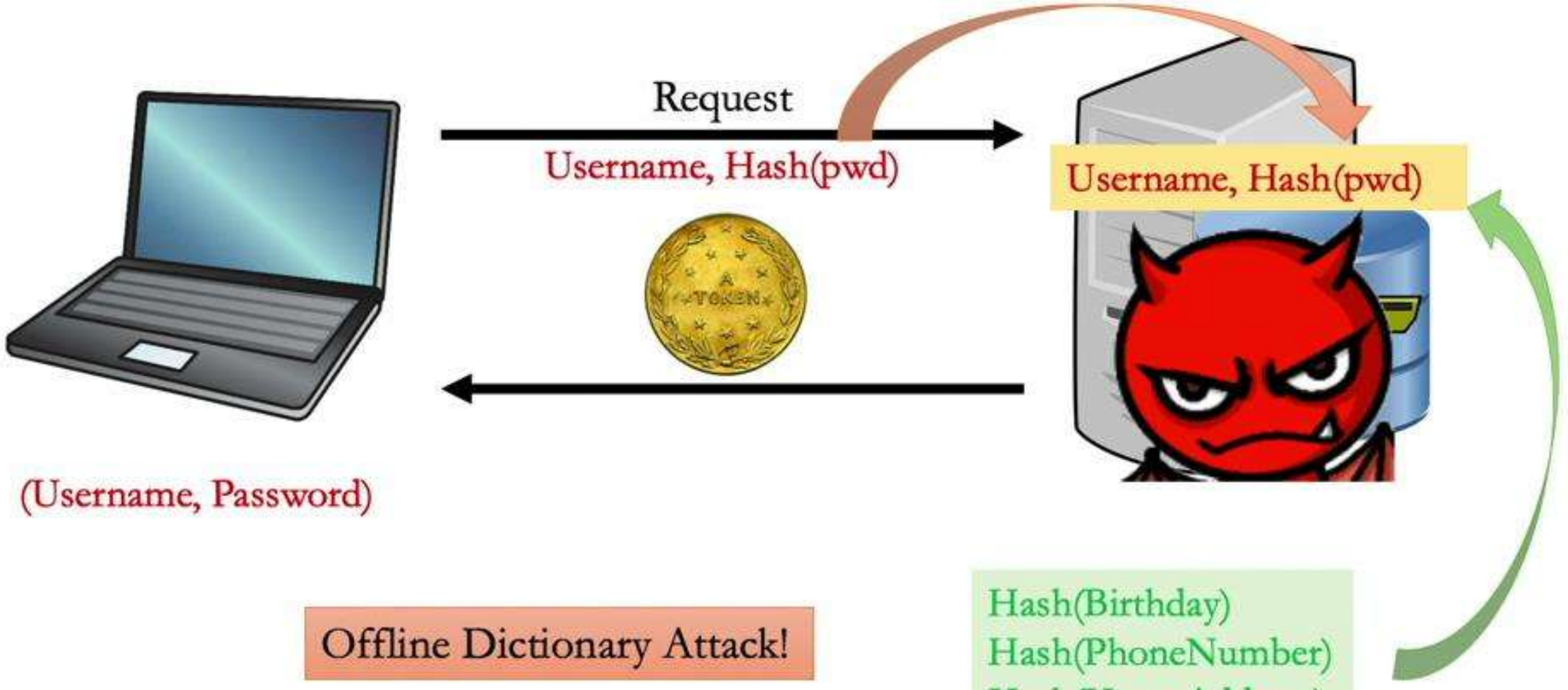


Hash(Birthday)
Hash(PhoneNumber)
Hash(HomeAddress)
⋮

Password Safety



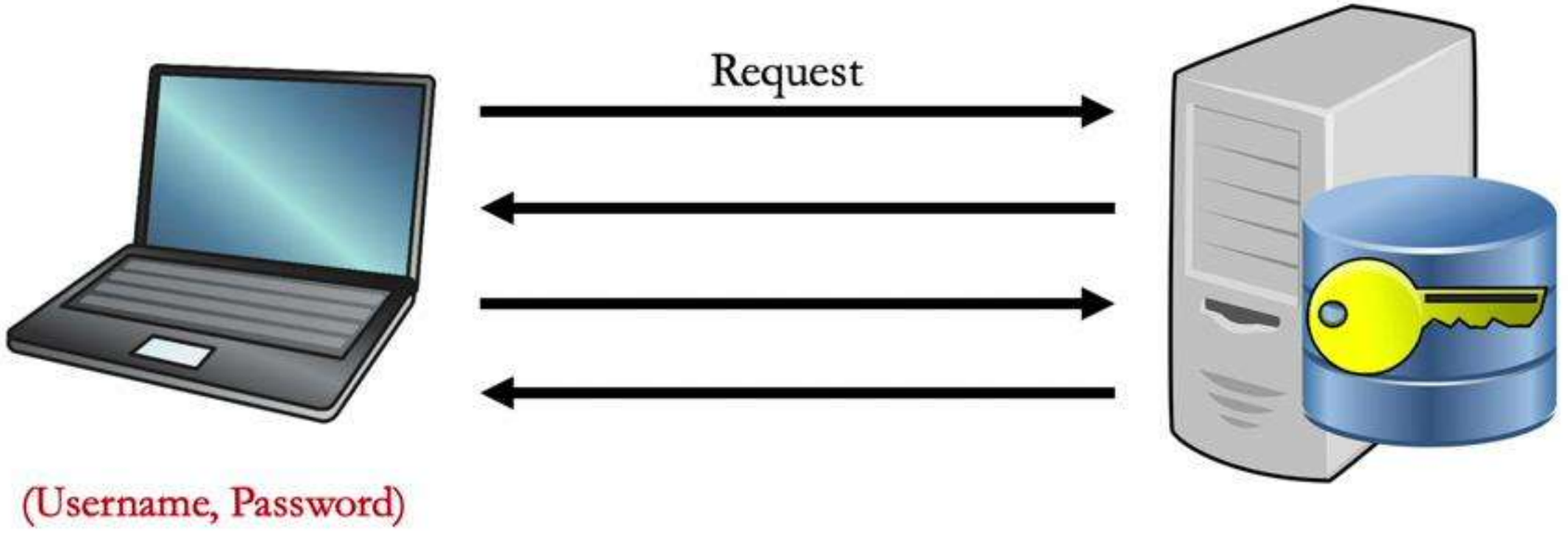
Password Safety



Password Safety

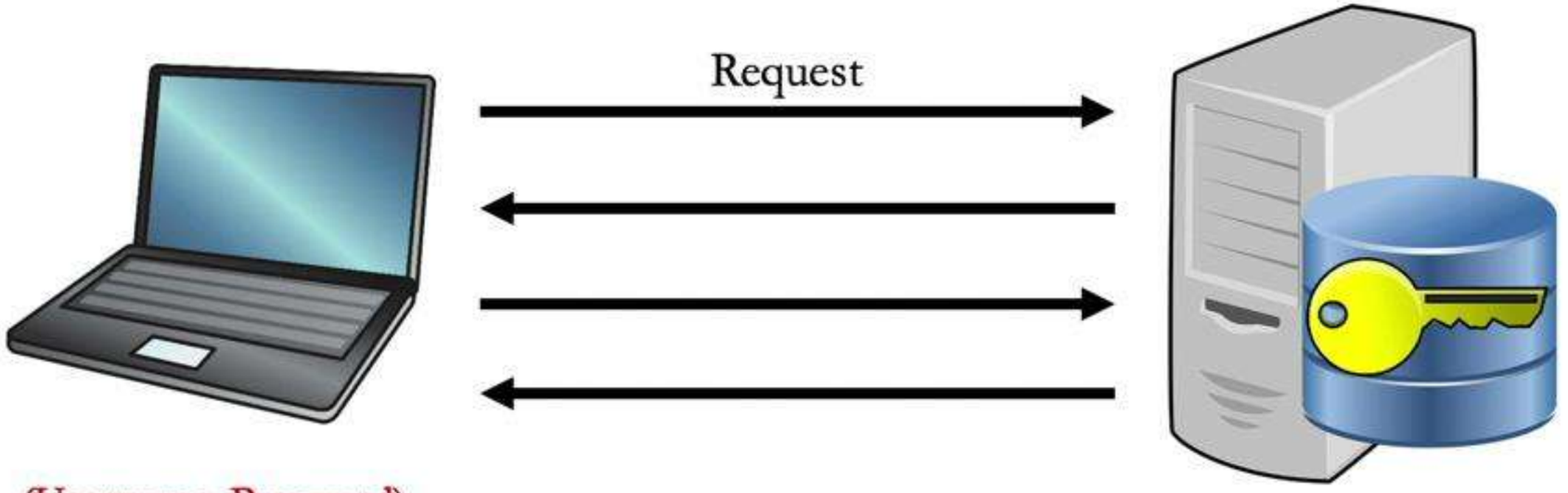
(Username, Password)

Password Safety

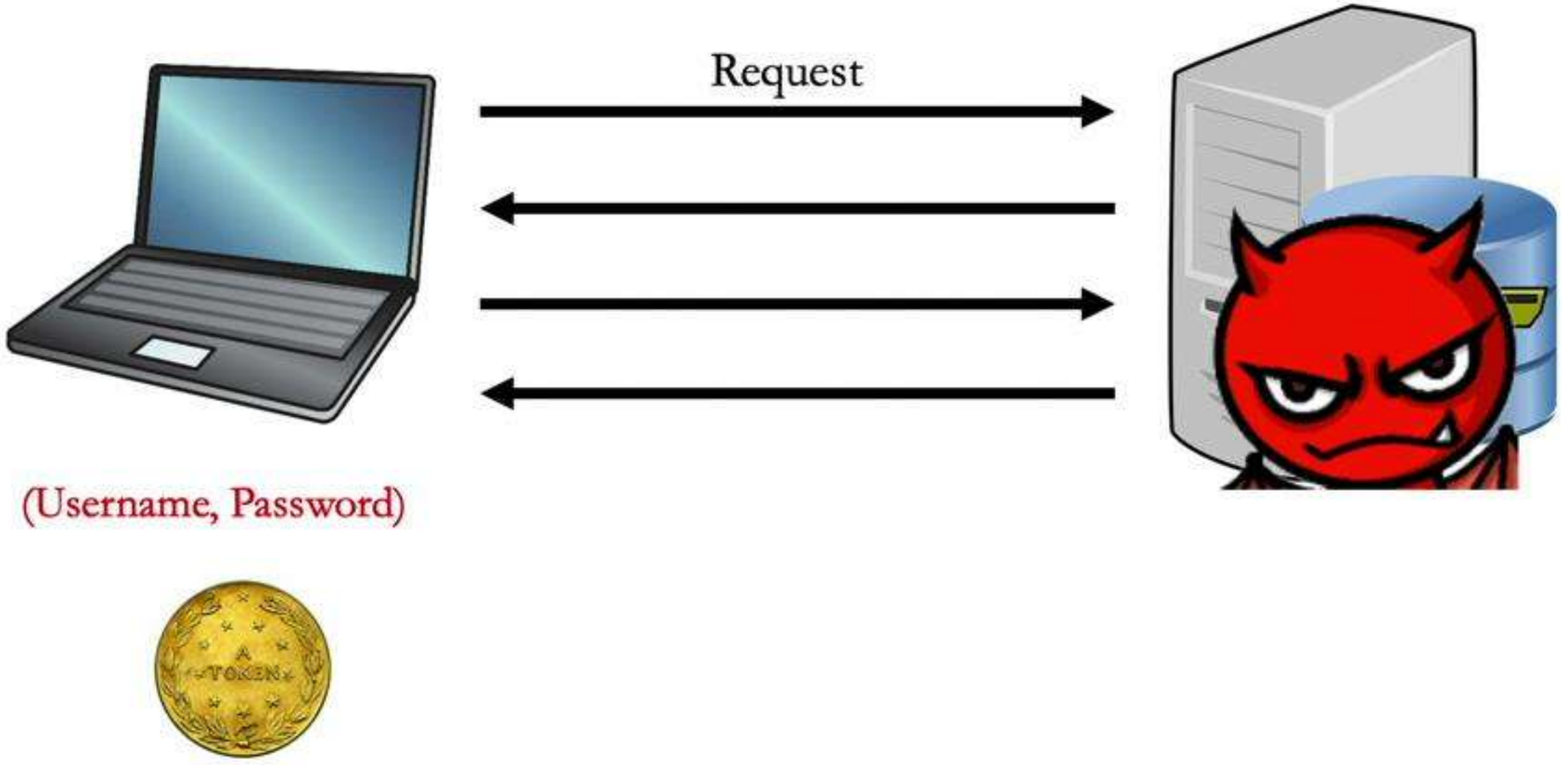


(Username, Password)

Password Safety

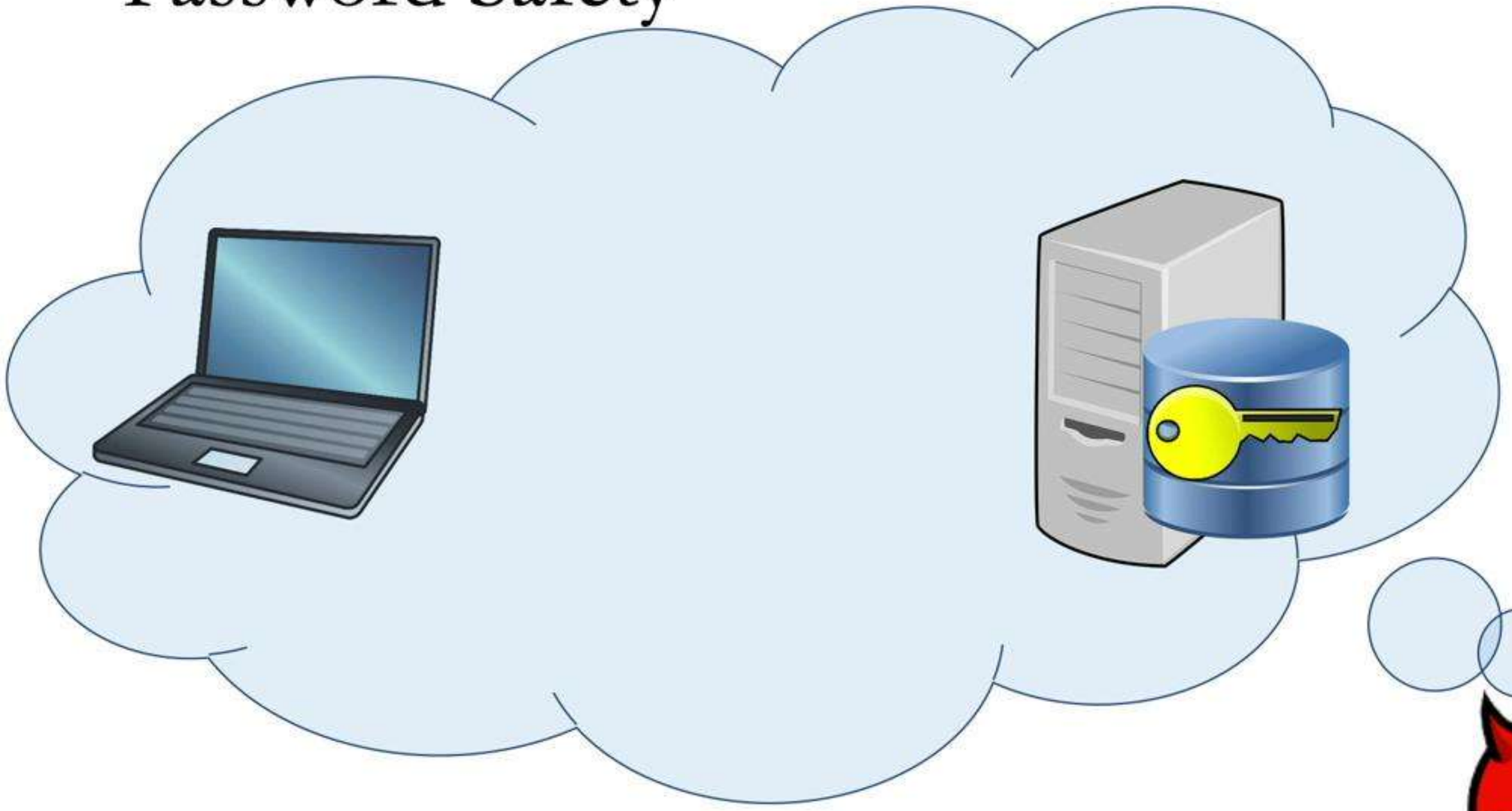


Password Safety

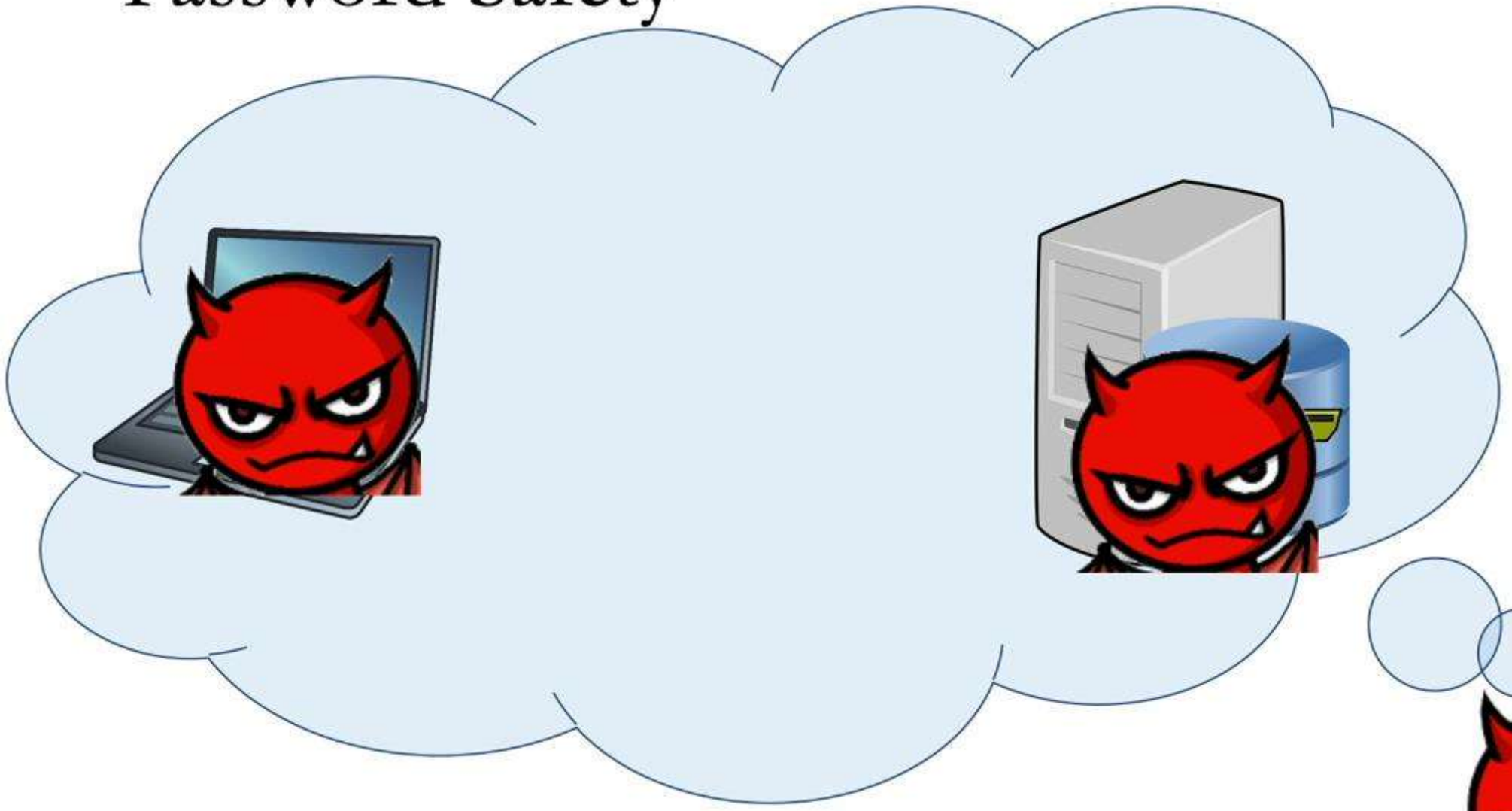


Password Safety

Password Safety



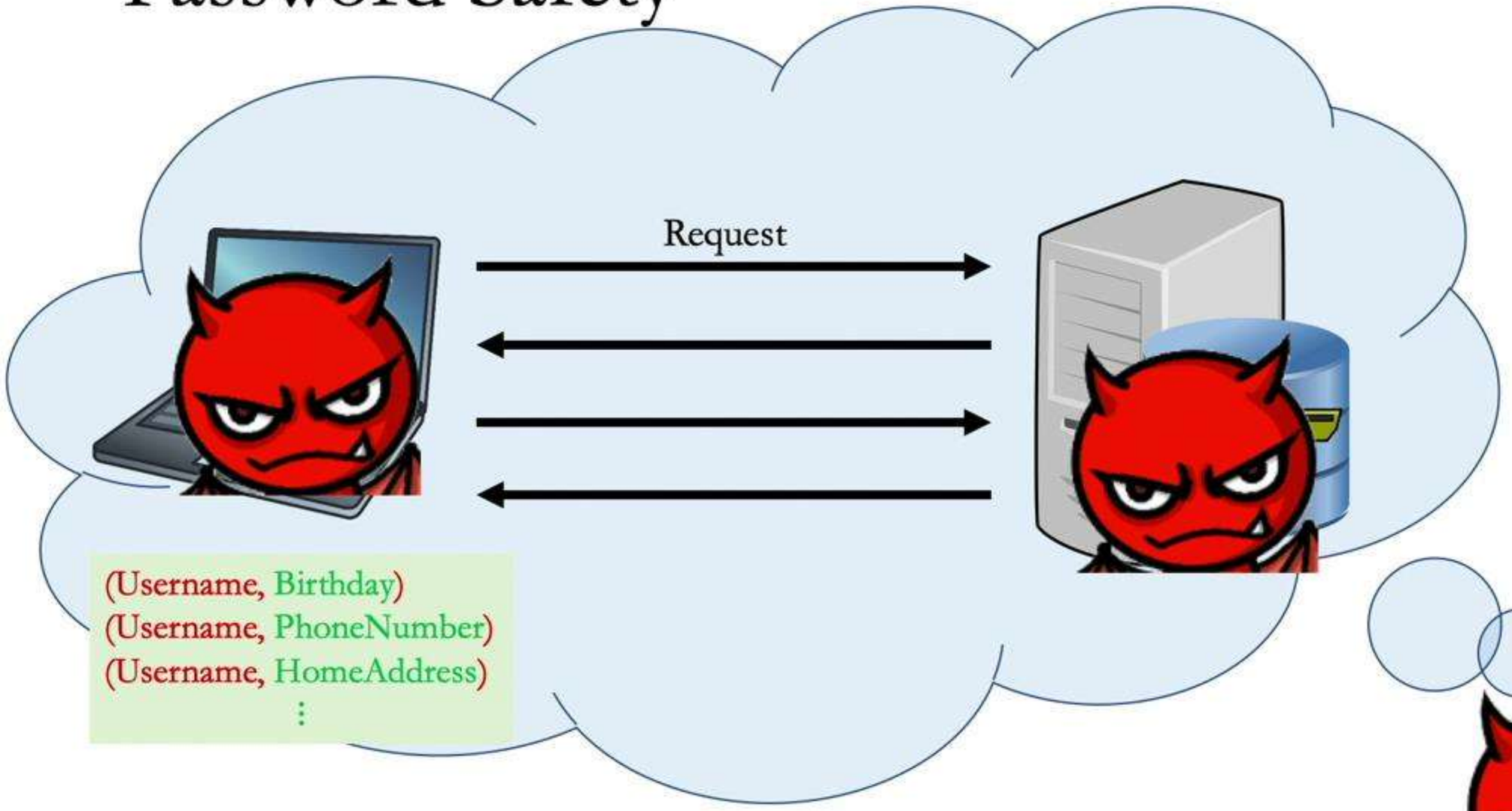
Password Safety



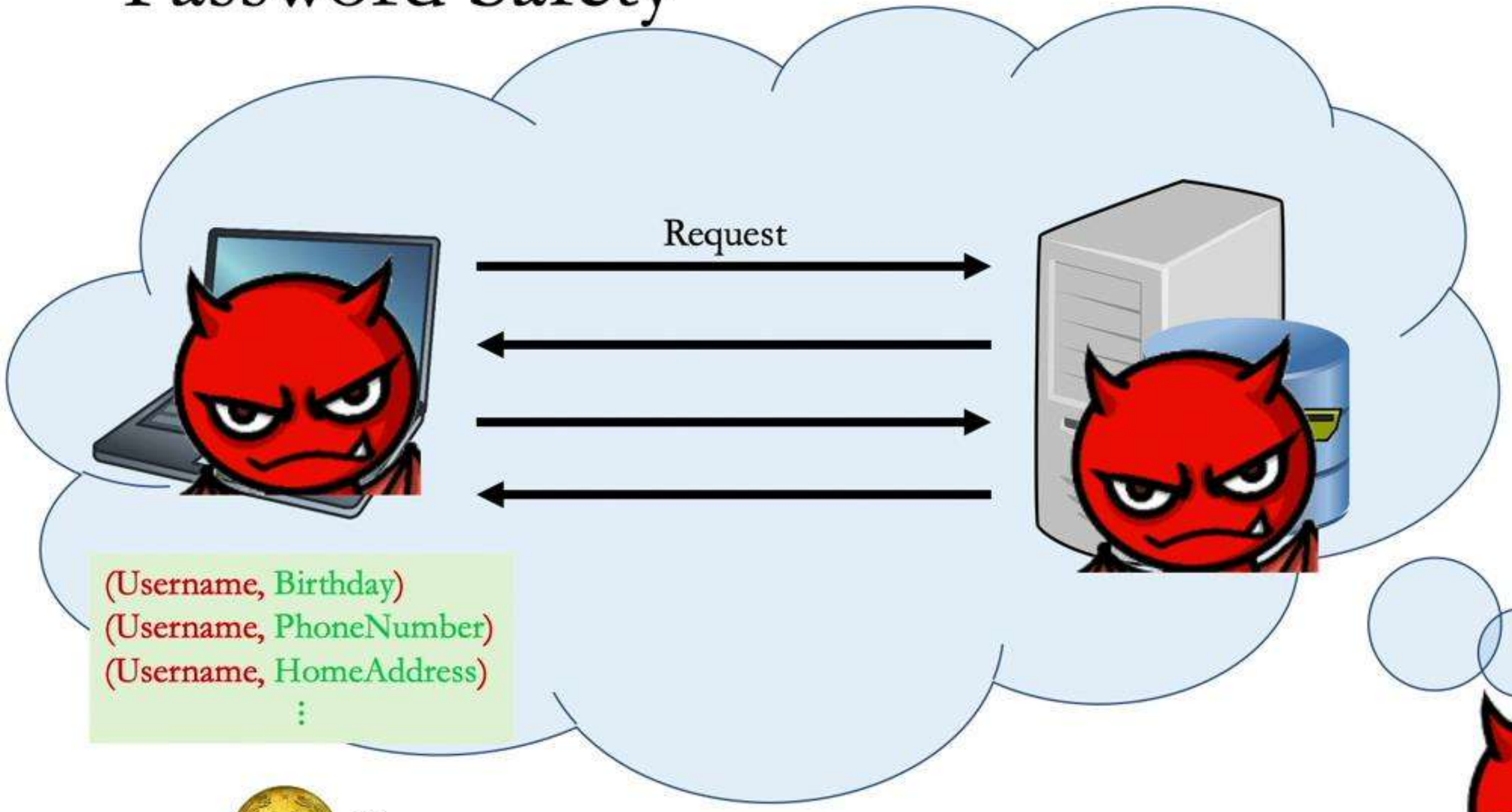
Password Safety

(Username, Birthday)
(Username, PhoneNumber)
(Username, HomeAddress)
⋮

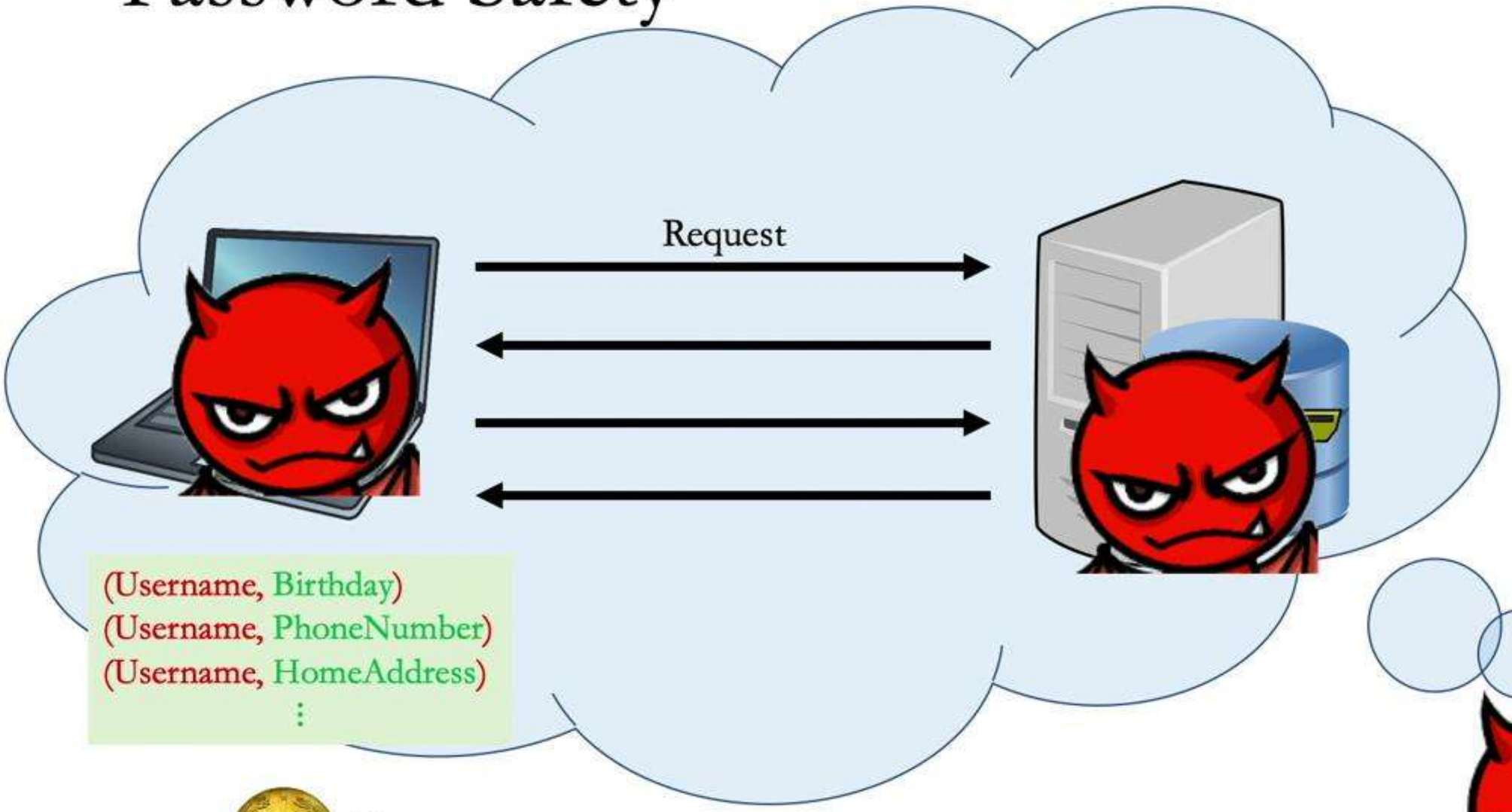
Password Safety



Password Safety



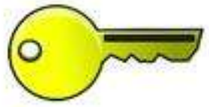
Password Safety



Our Goal

Our Goal

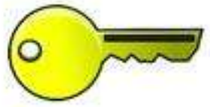
Our Goal



Our Goal

Our Goal

Our Goal



- If $< t$ servers are corrupted:

Our Goal

- If $< t$ servers are corrupted:
  - Secret key is secure

Our Goal

- If $< t$ servers are corrupted:
  - Secret key is secure
  - Password is secure (against offline attack)

Our Contribution

Our Contribution

- Password-based Threshold Authentication (PbTA)
  - Motivate the problem
  - Formalize a game-based security definition

Our Contribution

- Password-based Threshold Authentication (PbTA)
  - Motivate the problem
  - Formalize a game-based security definition

- **PASTA**: a framework of construction for various types of tokens
  - Minimal round complexity
  - Low computational & communication complexity

Our Contribution

- Password-based Threshold Authentication (PbTA)
  - Motivate the problem
  - Formalize a game-based security definition

- **PASTA**: a framework of construction for various types of tokens
  - Minimal round complexity
  - Low computational & communication complexity

- Implementations and experiments

Our Contribution

- Password-based Threshold Authentication (PbTA)
  - Motivate the problem
  - Formalize a game-based security definition

- **PASTA**: a framework of construction for various types of tokens
  - Minimal round complexity
  - Low computational & communication complexity

- Implementations and experiments

Global Setup

Global Setup

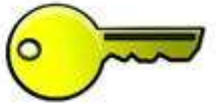
Global Setup

Global Setup

Any t servers can recover the secret key.

(In this example, $t = 3$.)

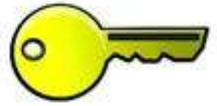
Global Setup



Any t servers can recover the secret key.

(In this example, $t = 3$.)

Global Setup



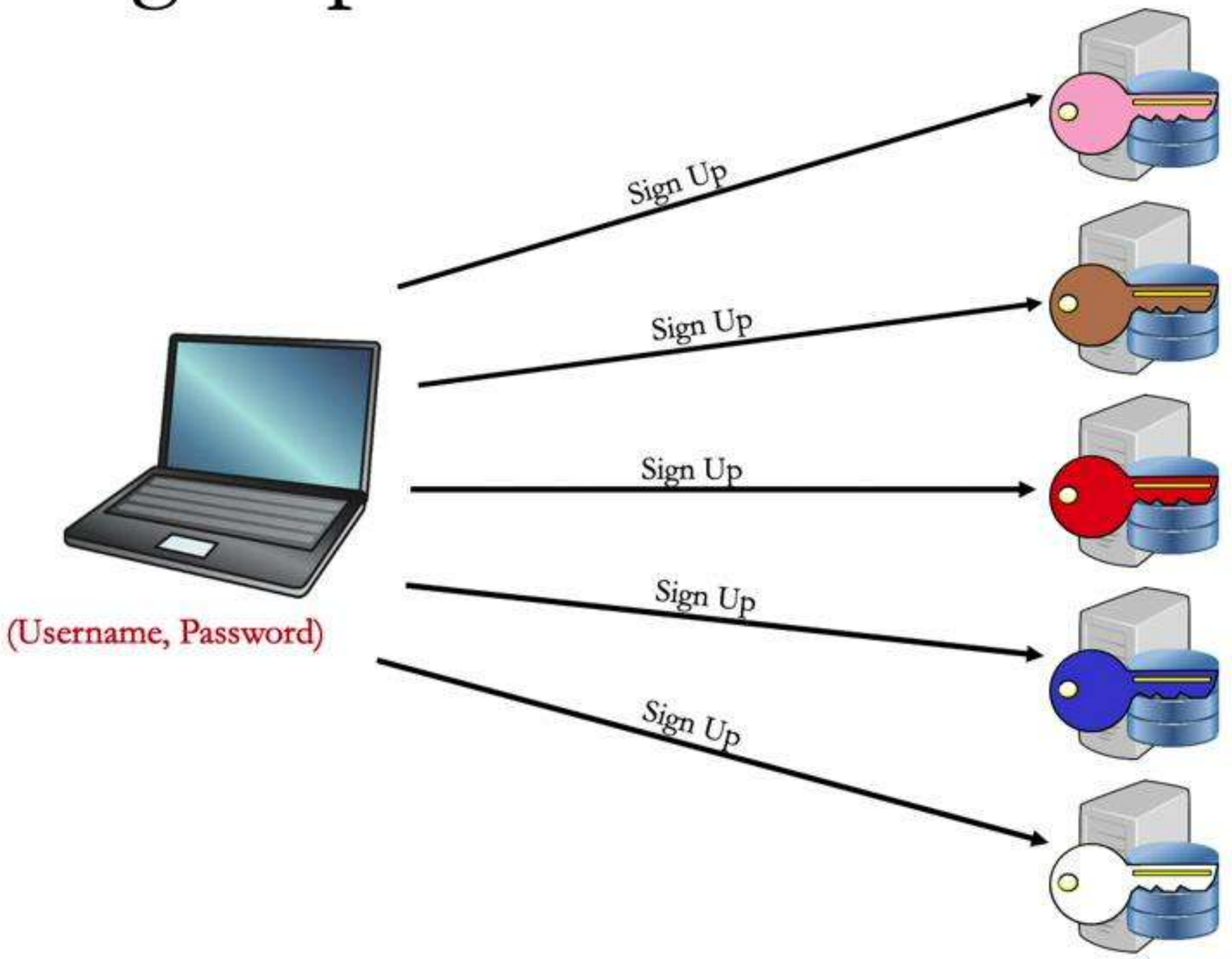
Any t servers can recover the secret key.

Any $t-1$ servers cannot recover the secret key.

(In this example, $t = 3$.)

User Sign Up

User Sign Up

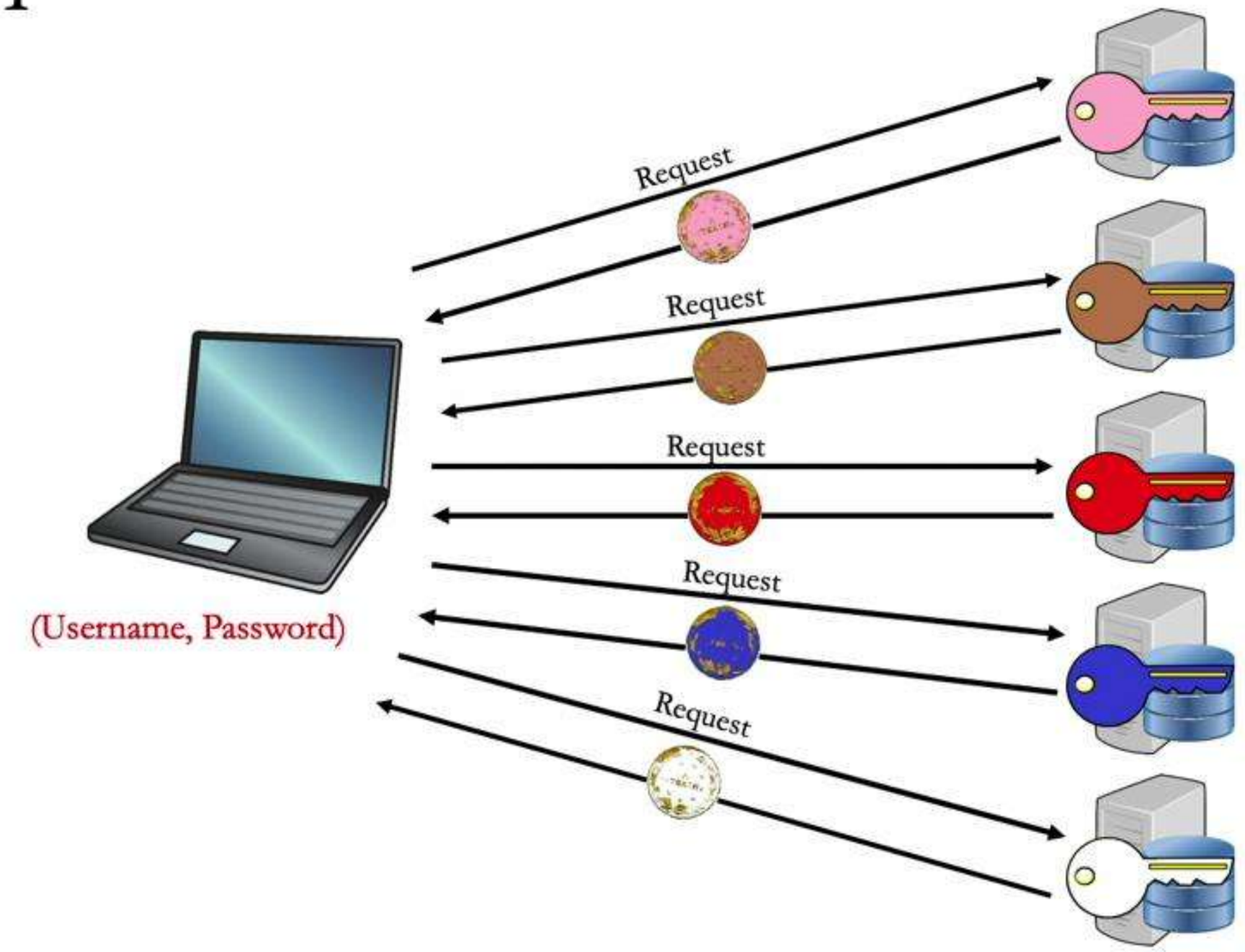


Request for Token

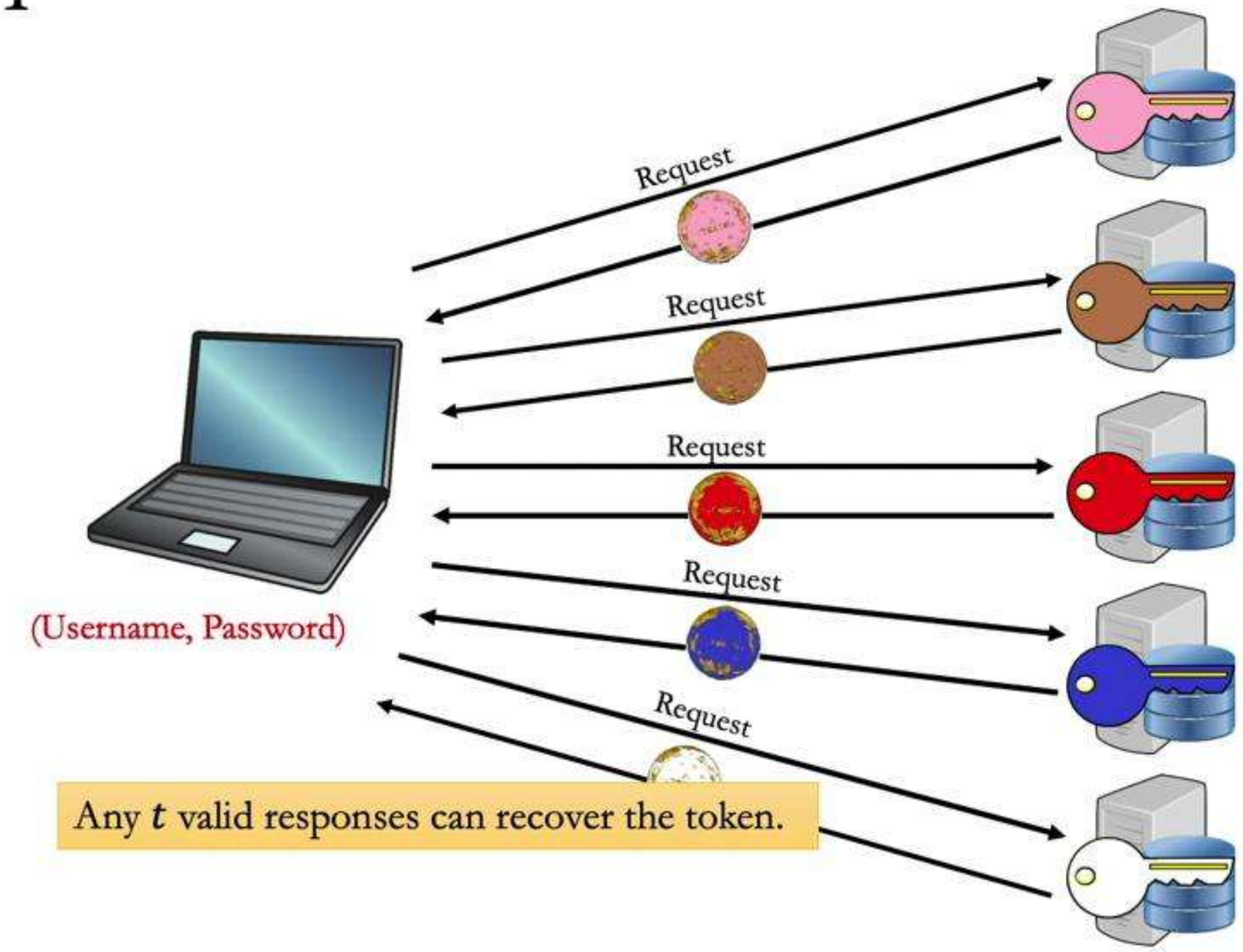
Request for Token

(Username, Password)

Request for Token



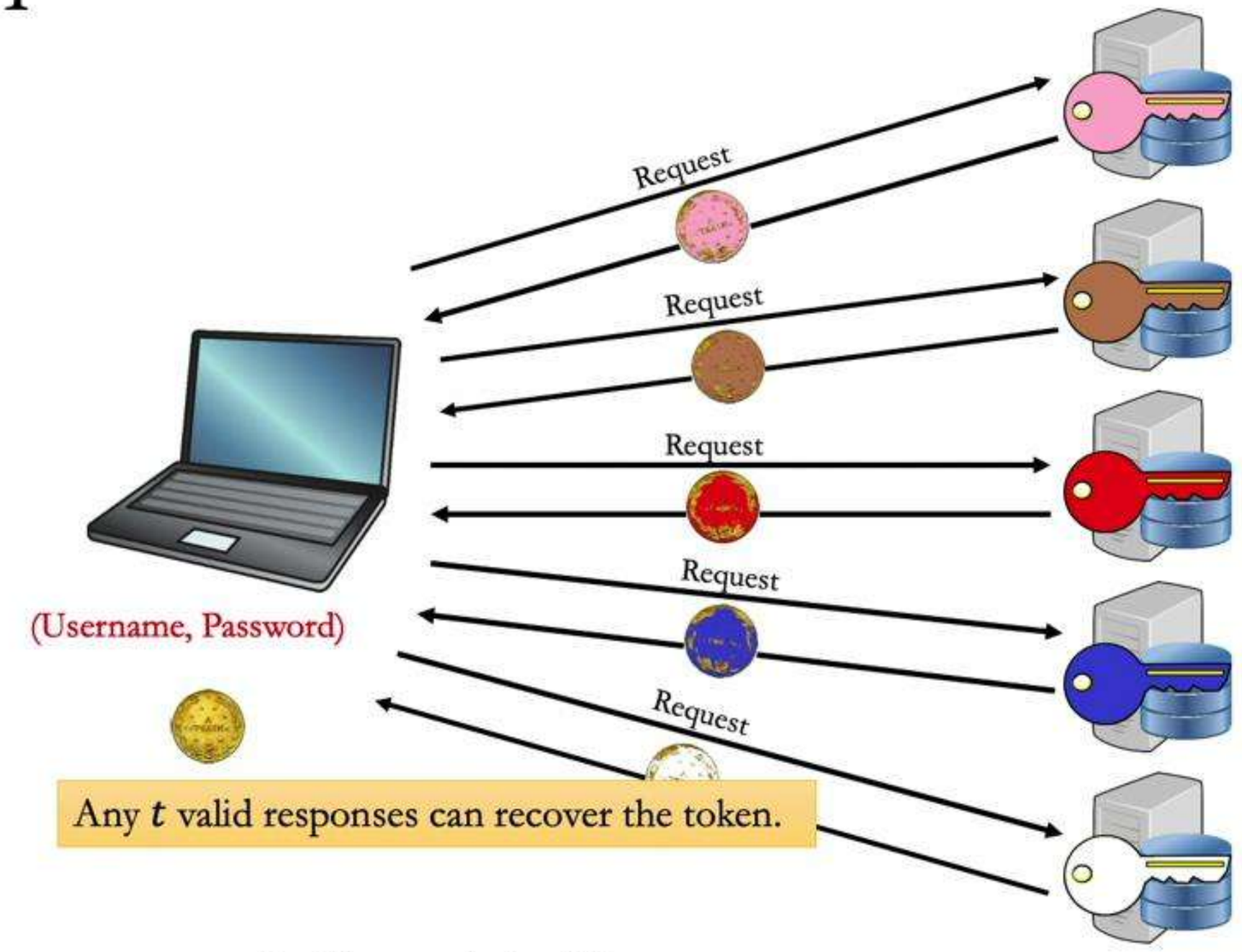
Request for Token



Any t valid responses can recover the token.

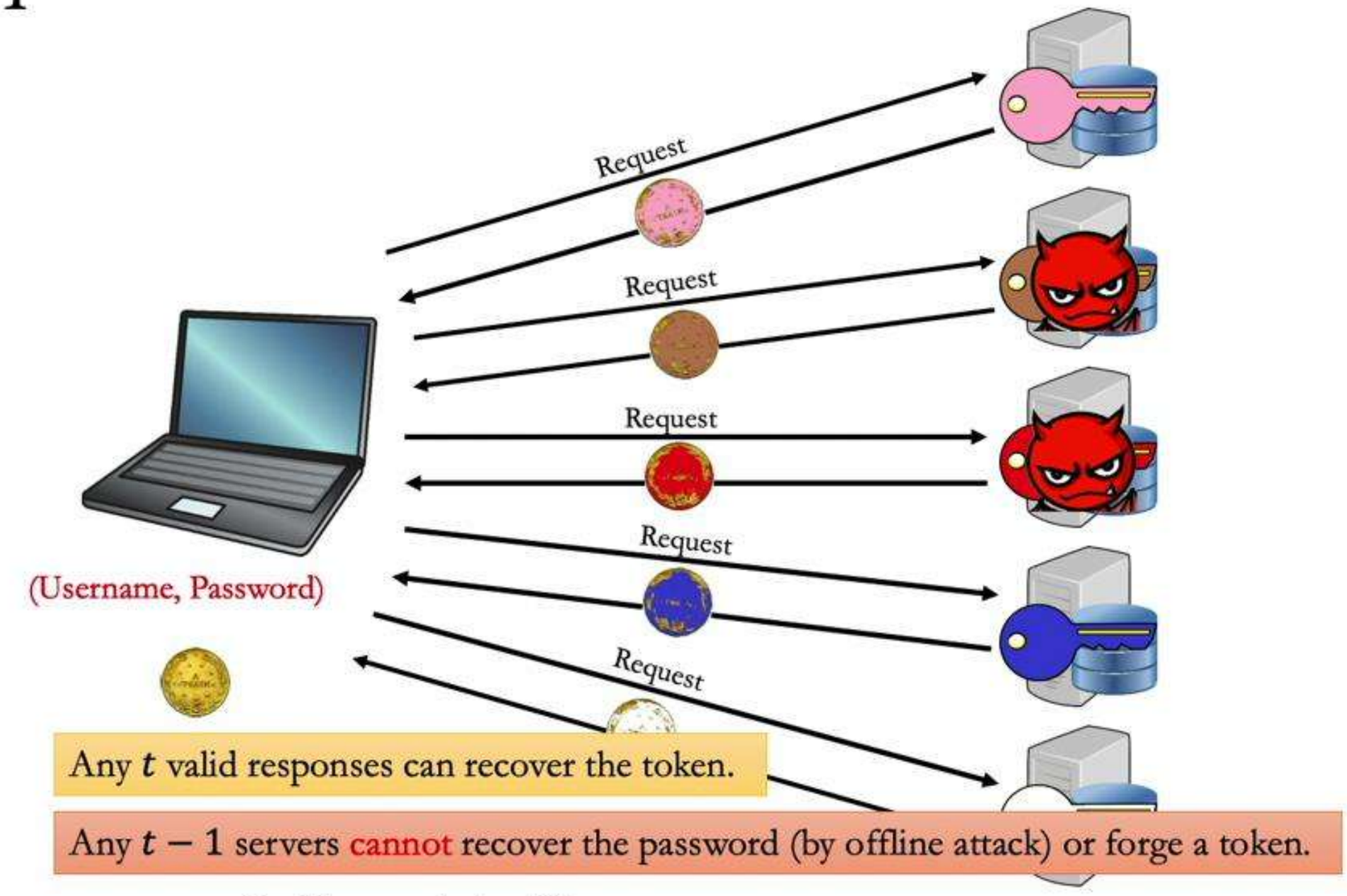
(In this example, $t = 3$.)

Request for Token



(In this example, $t = 3$.)

Request for Token



Any t valid responses can recover the token.

Any $t-1$ servers cannot recover the password (by offline attack) or forge a token.

(In this example, $t = 3$.)

Overview

- What problem is PASTA trying to solve?
- How does PASTA work?
- Is it practical?

Our Approach

Our Approach

- Non-Interactive Threshold Token Generation (NITTG)
  - Symmetric key based MAC [NPR'99, MPSWW'02]
  - Public key (DDH) based MAC [NPR'99]
  - RSA based digital signature [Shoup'00]
  - Pairing based digital signature [BLS'01, Boldyreva'03]

- Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]
  - Formalize game-based definition for our needs
  - Prove security for the construction

Our Approach

- Non-Interactive Threshold Token Generation (NITTG)
  - Symmetric key based MAC [NPR'99, MPSWW'02]
  - Public key (DDH) based MAC [NPR'99]
  - RSA based digital signature [Shoup'00]
  - Pairing based digital signature [BLS'01, Boldyreva'03]

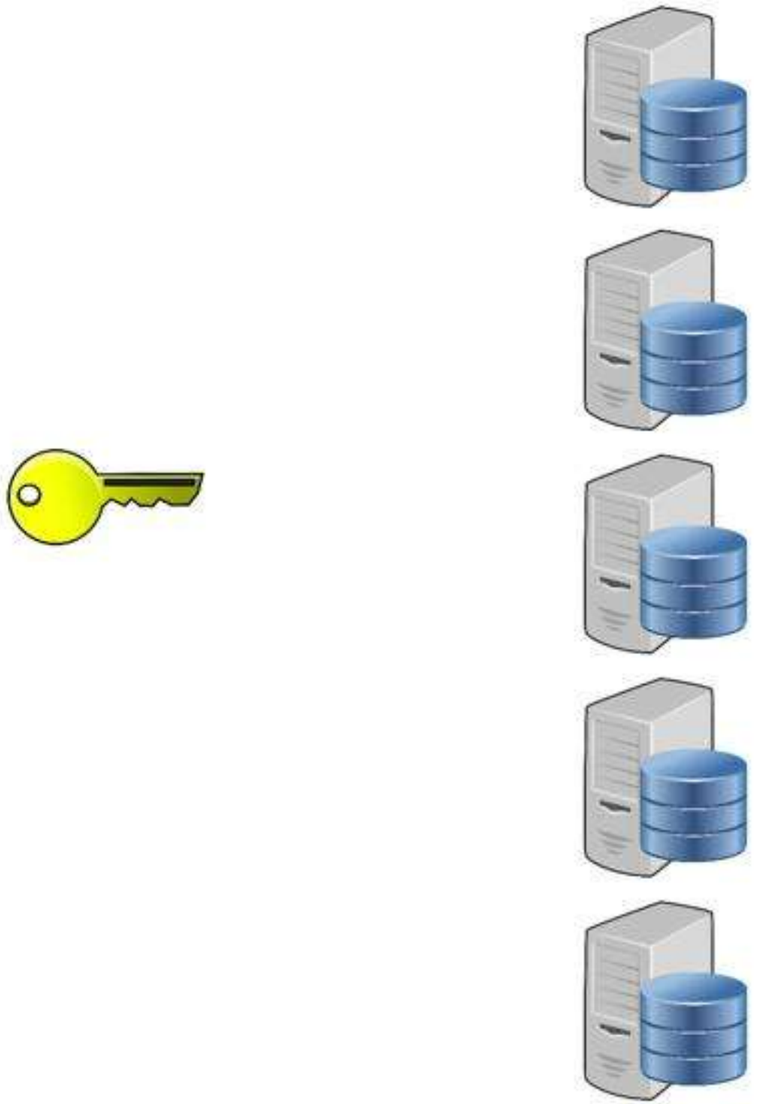
- Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]
  - Formalize game-based definition for our needs
  - Prove security for the construction

- Generic construction of PASTA from NITTG + TOPRF

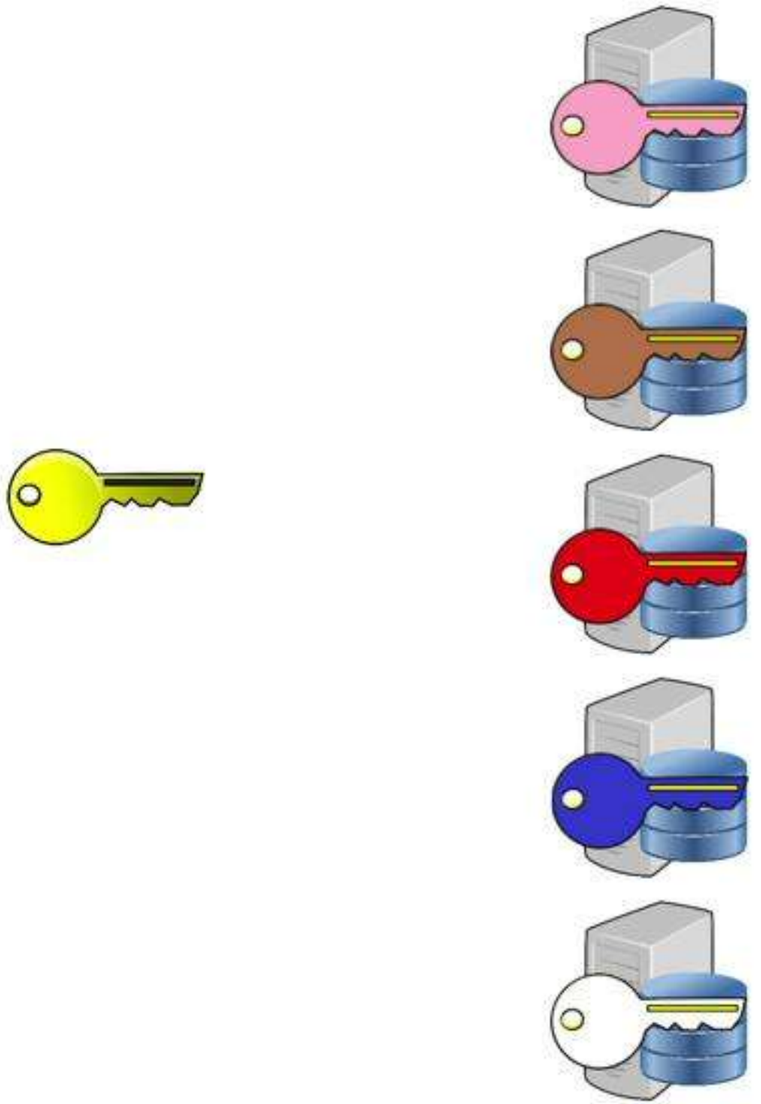
Non-Interactive Threshold Token Generation (NITTG)

Non-Interactive Threshold Token Generation (NITTG)

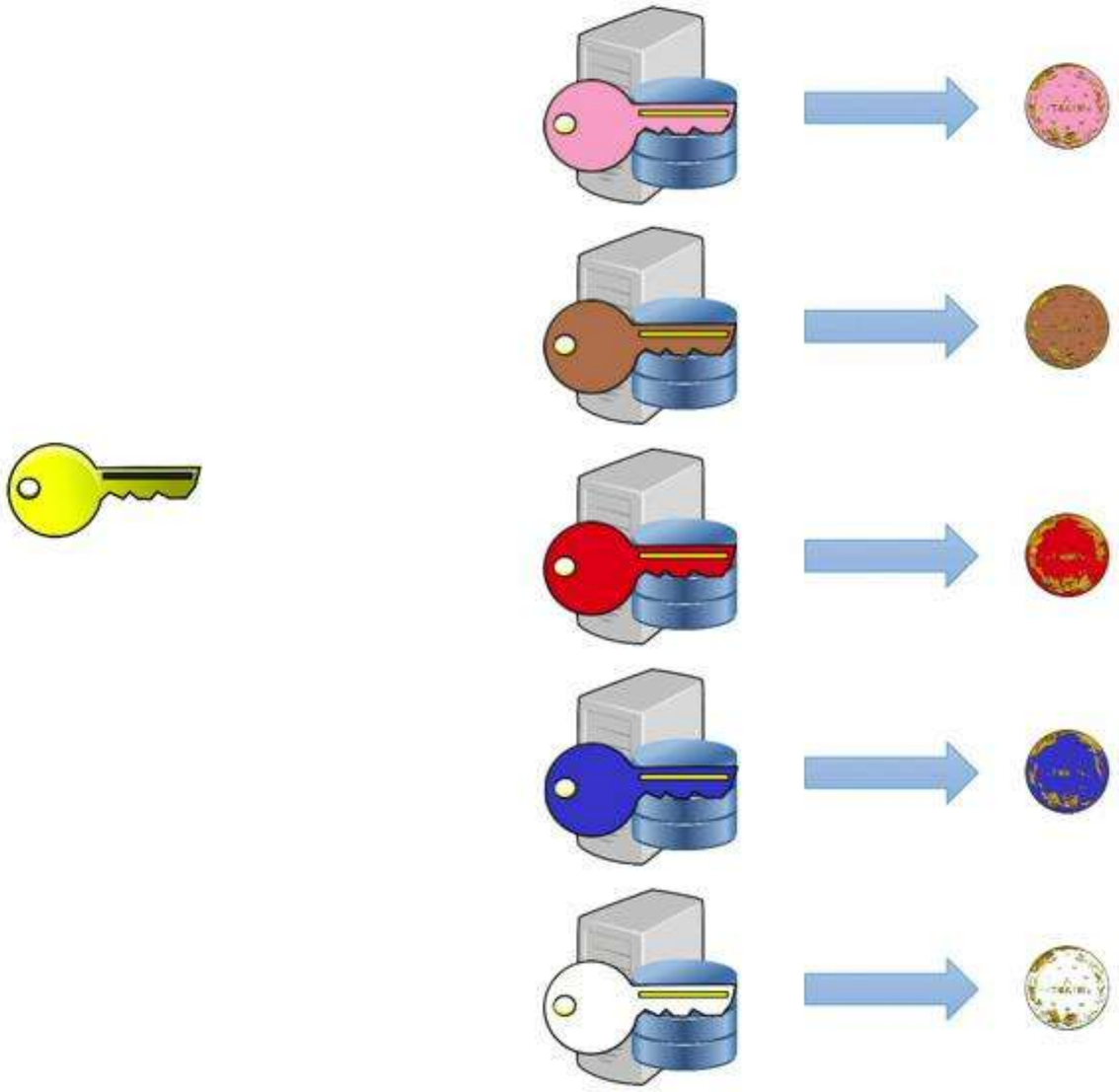
Non-Interactive Threshold Token Generation (NITTG)



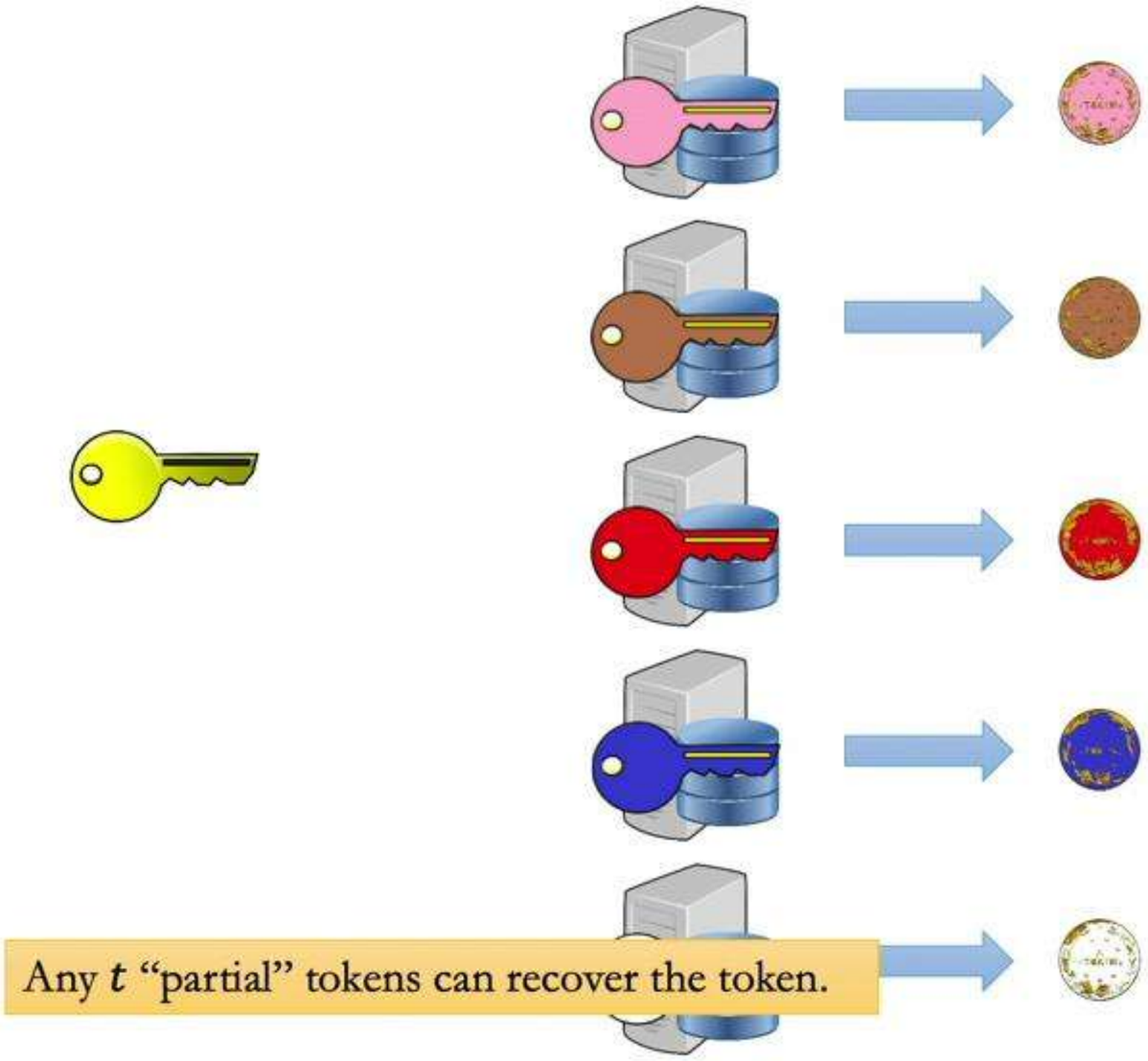
Non-Interactive Threshold Token Generation (NITTG)



Non-Interactive Threshold Token Generation (NITTG)

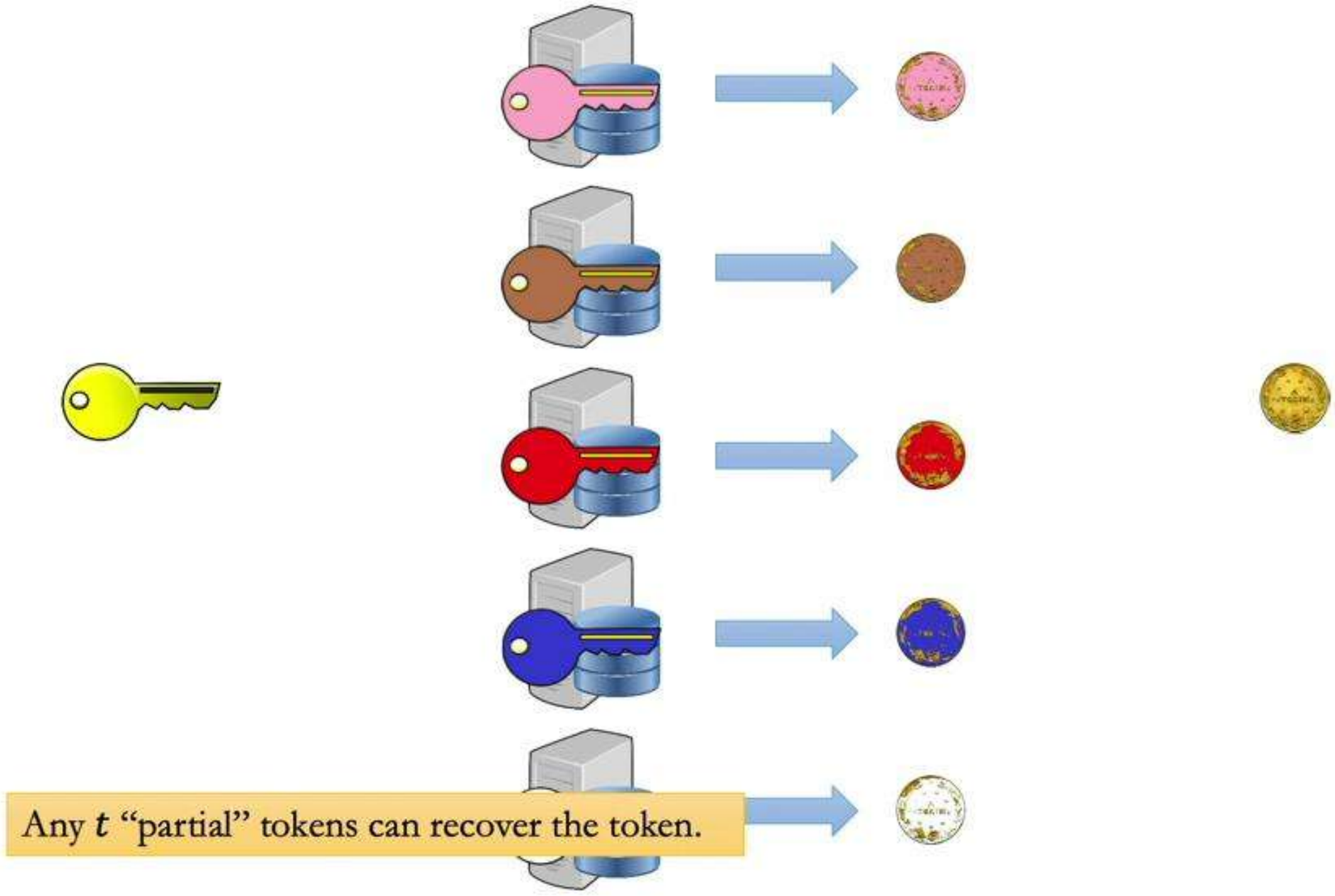


Non-Interactive Threshold Token Generation (NITTG)



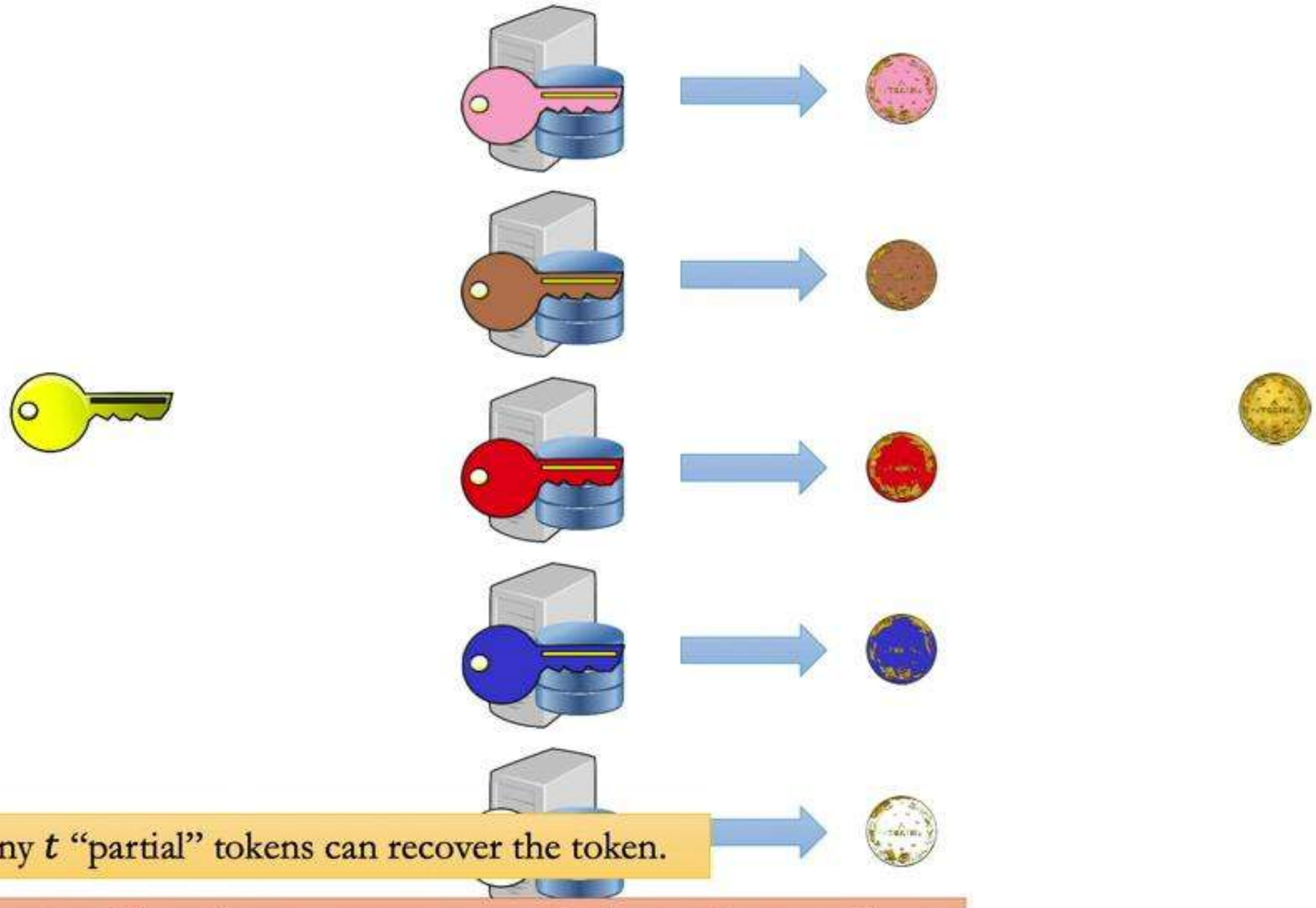
(In this example, $t = 3$.)

Non-Interactive Threshold Token Generation (NITTG)



(In this example, $t = 3$.)

Non-Interactive Threshold Token Generation (NITTG)

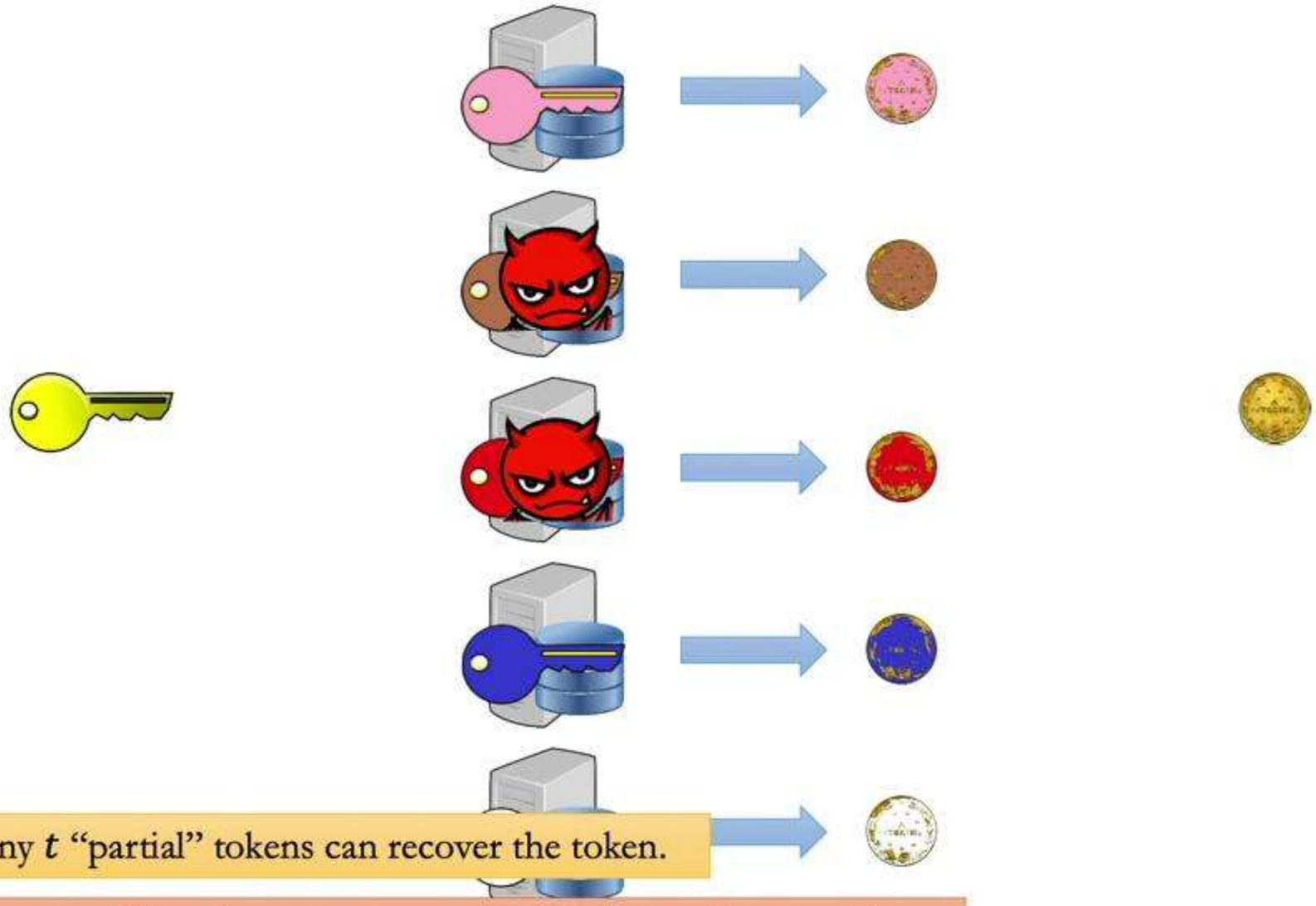


Any t "partial" tokens can recover the token.

Any $t - 1$ key shares cannot recover the key or forge a token.

(In this example, $t = 3$.)

Non-Interactive Threshold Token Generation (NITTG)



Any t "partial" tokens can recover the token.

Any $t - 1$ key shares cannot recover the key or forge a token.

(In this example, $t = 3$.)

Naïve Solution

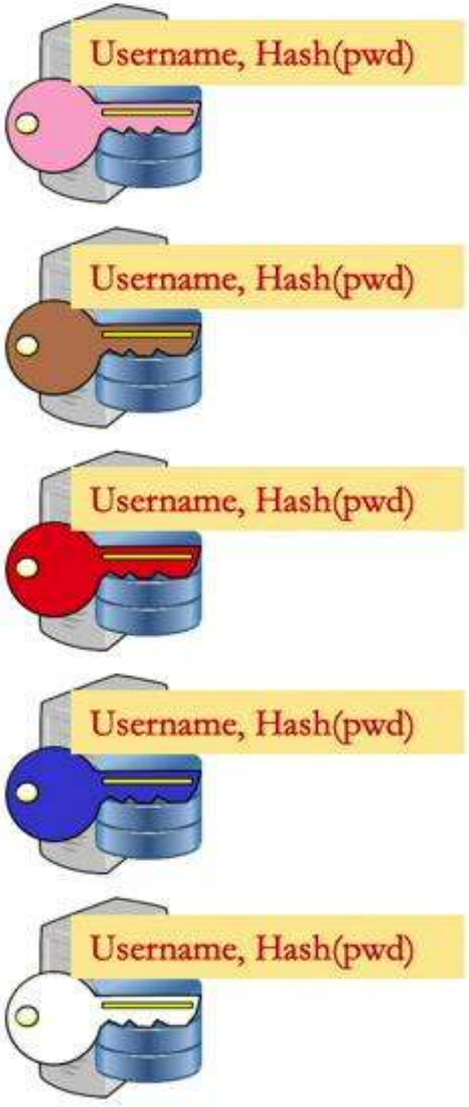
(Username, Password)

Naïve Solution

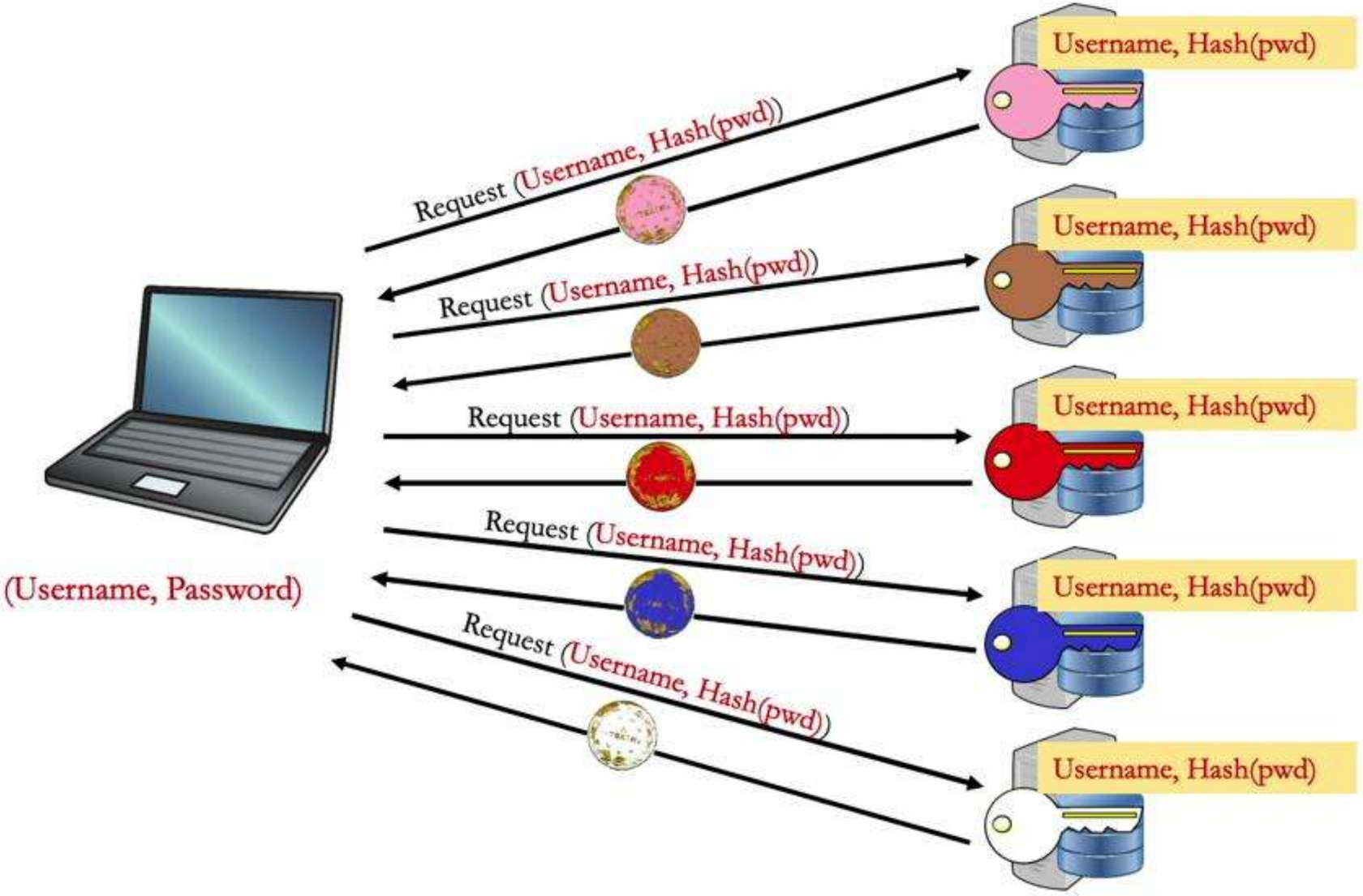
(Username, Password)

Naïve Solution

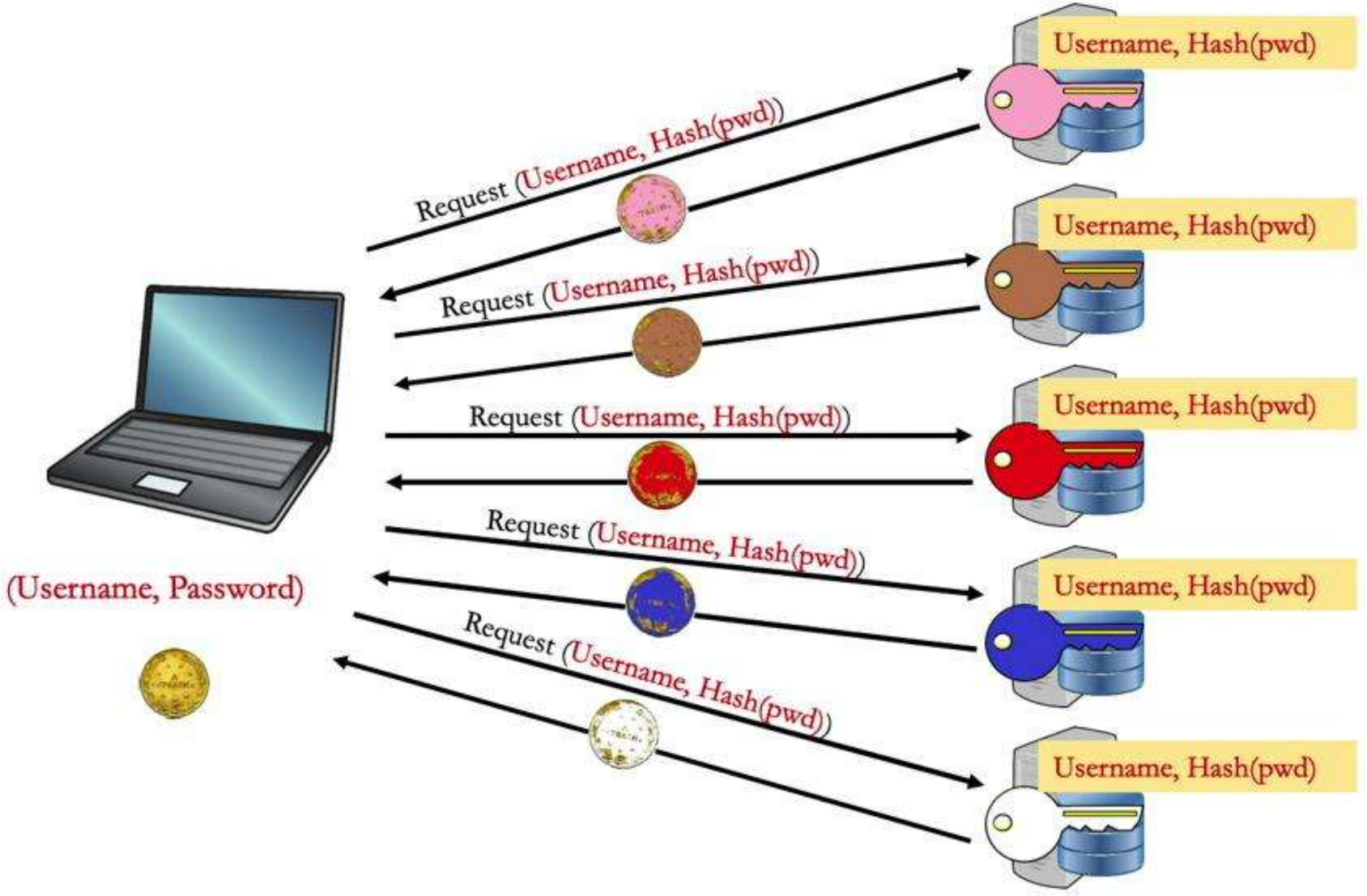
(Username, Password)



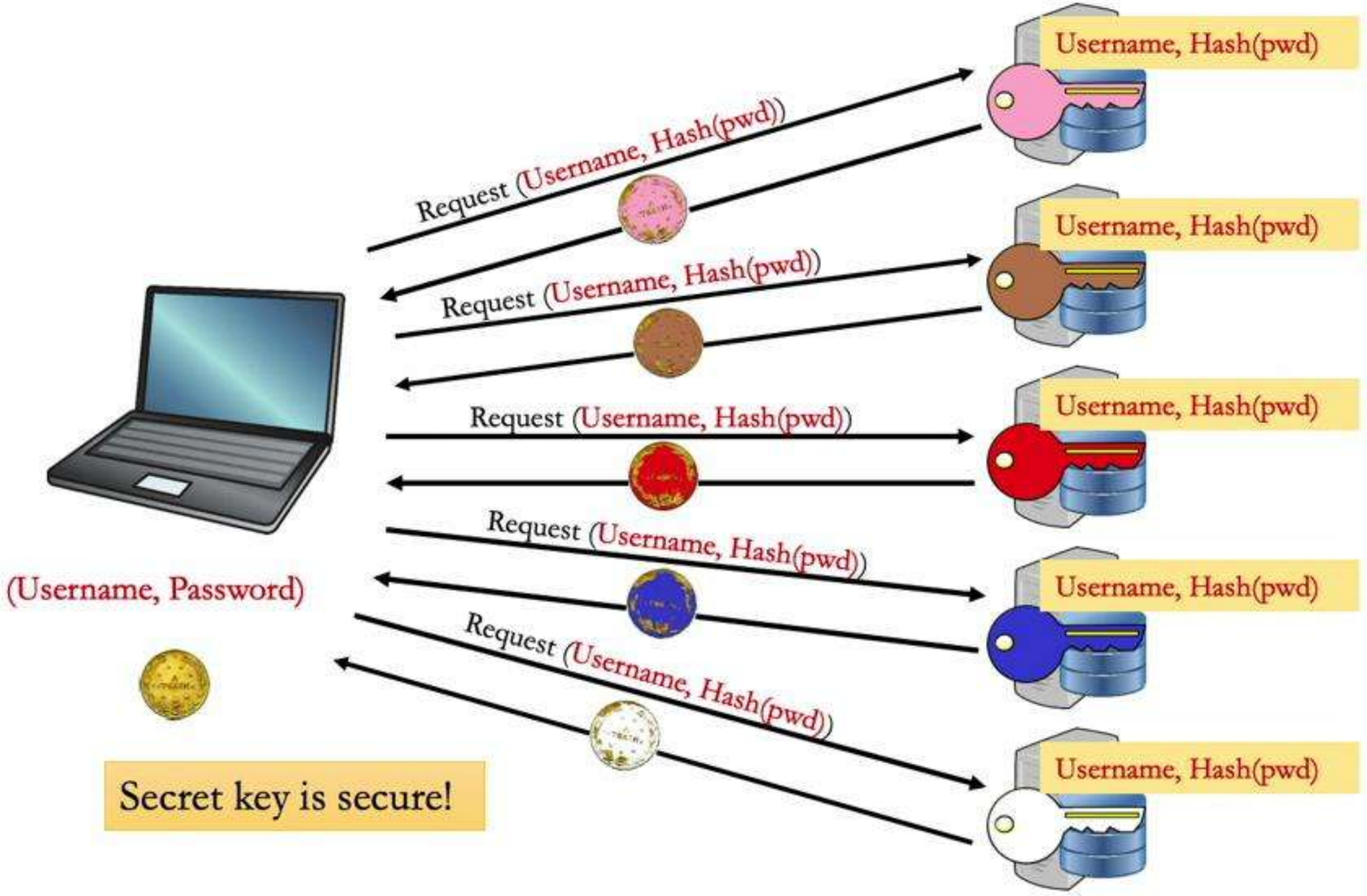
Naïve Solution



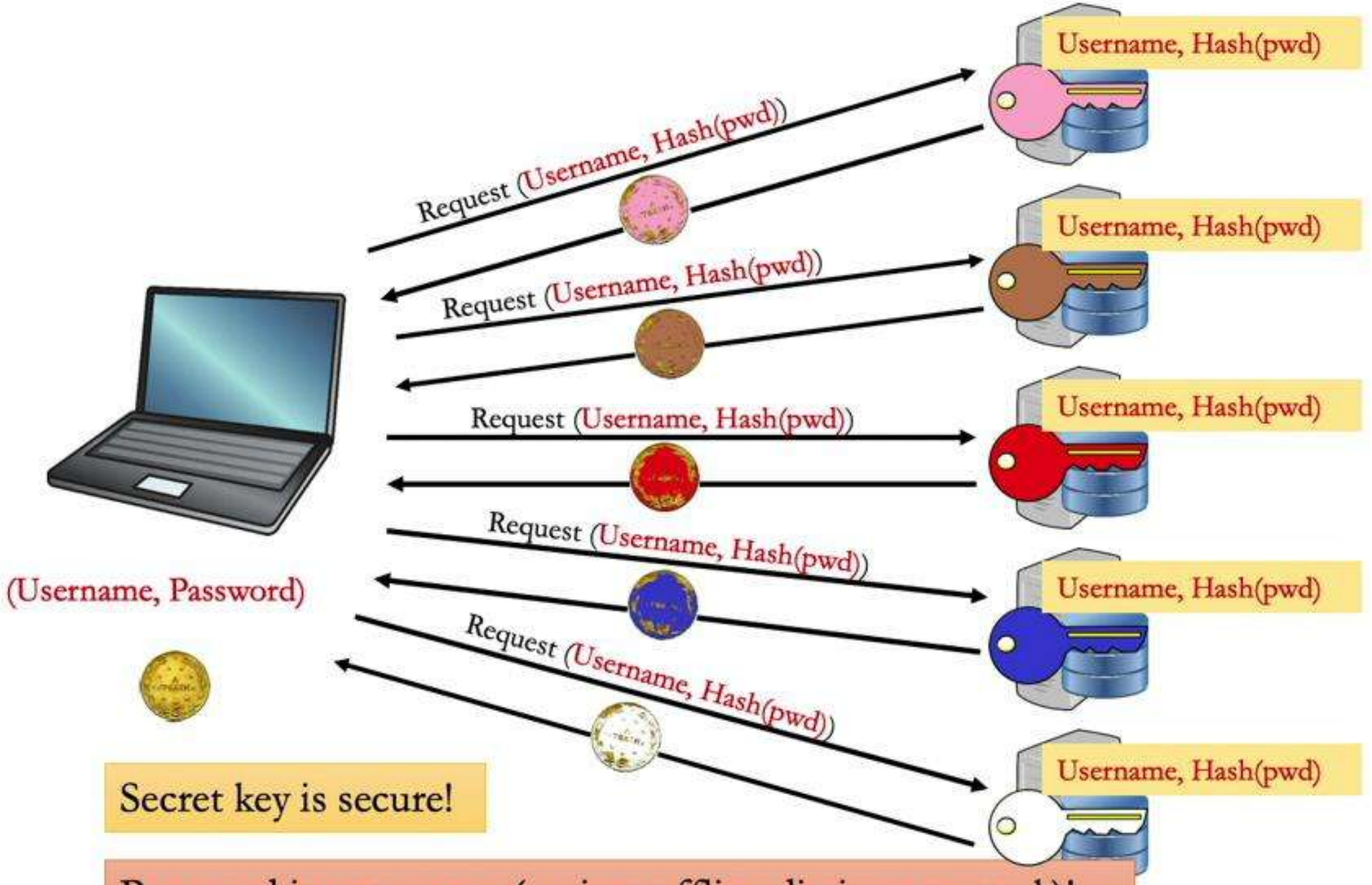
Naïve Solution



Naïve Solution



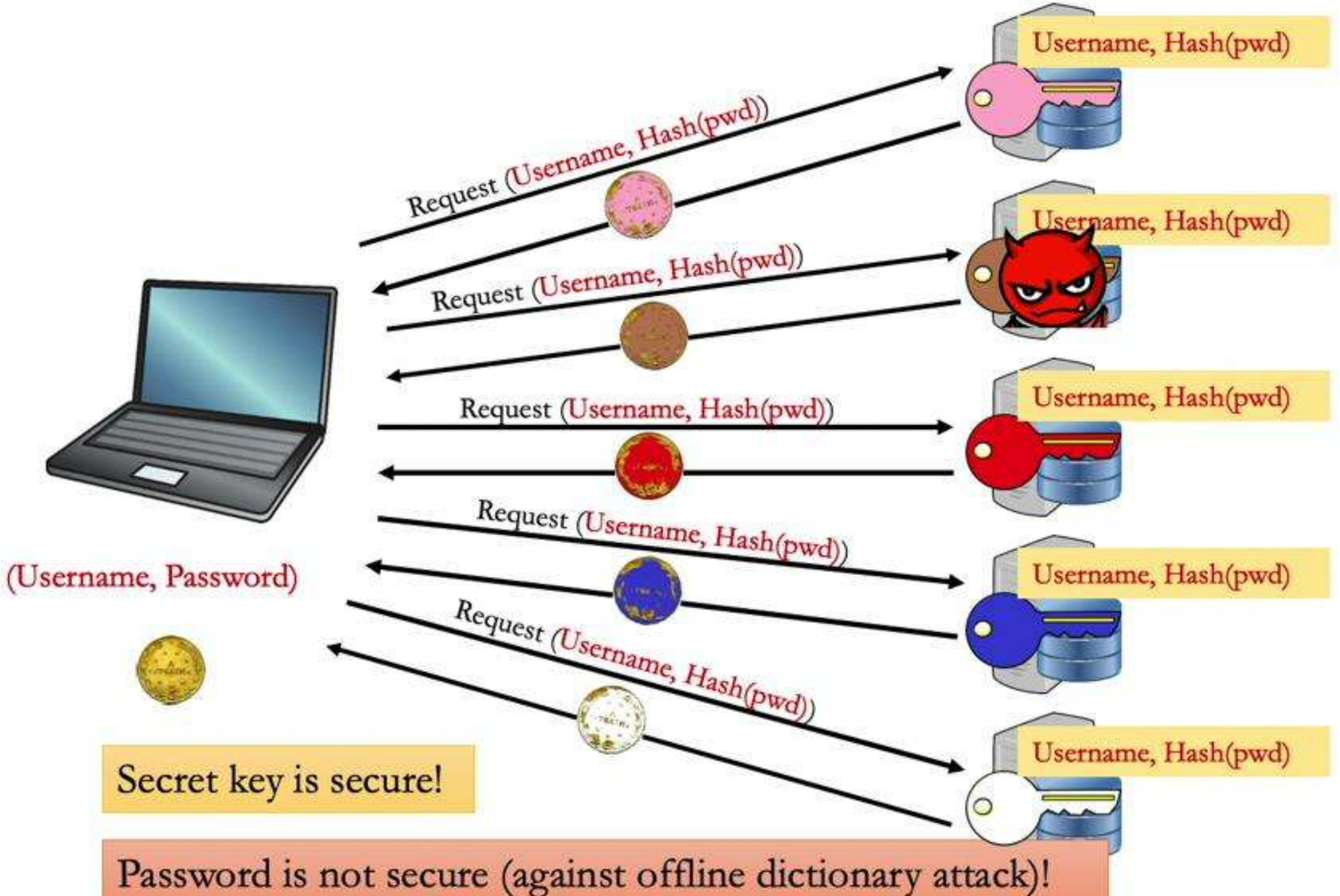
Naïve Solution



Secret key is secure!

Password is not secure (against offline dictionary attack)!

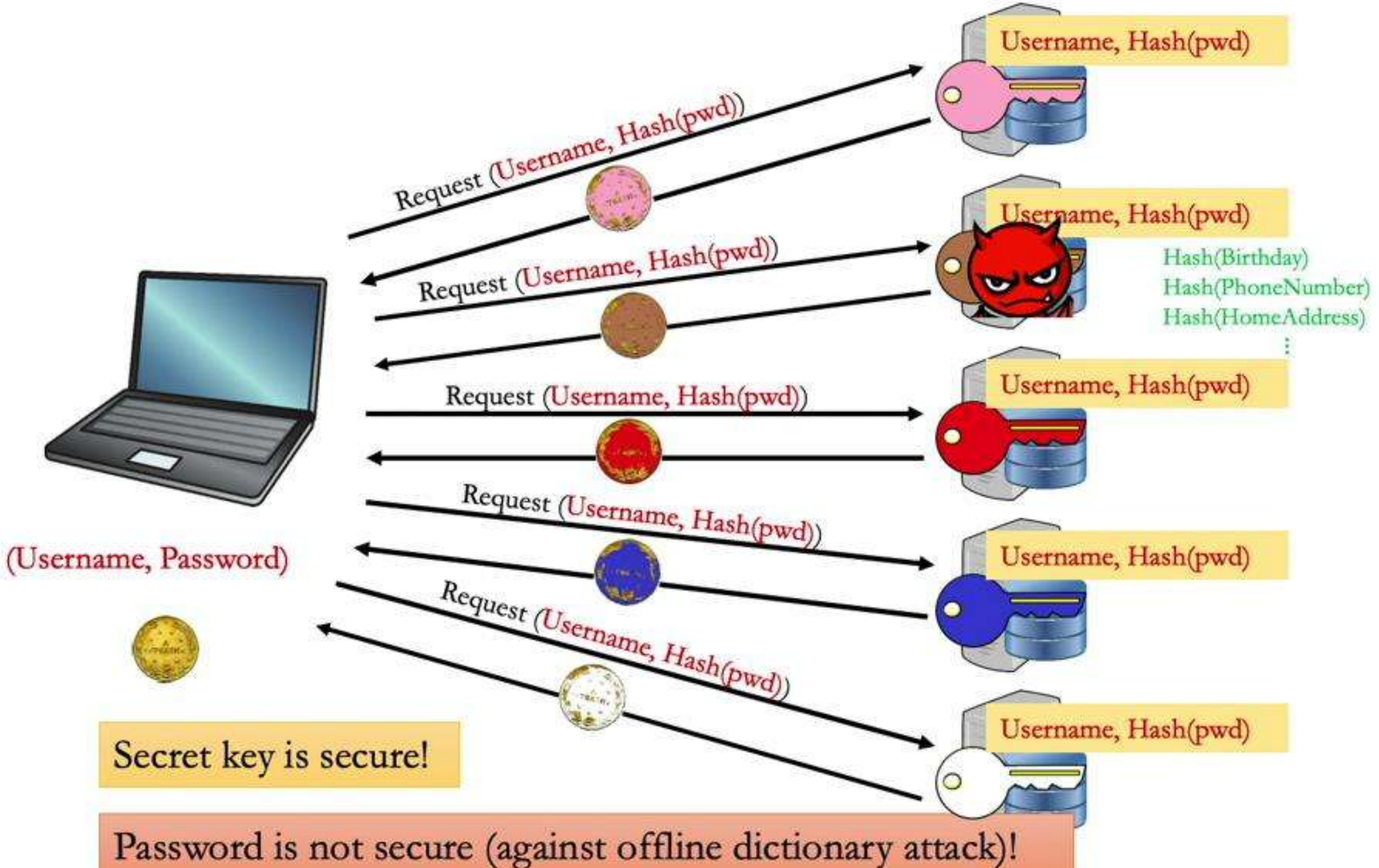
Naïve Solution



Secret key is secure!

Password is not secure (against offline dictionary attack)!

Naïve Solution



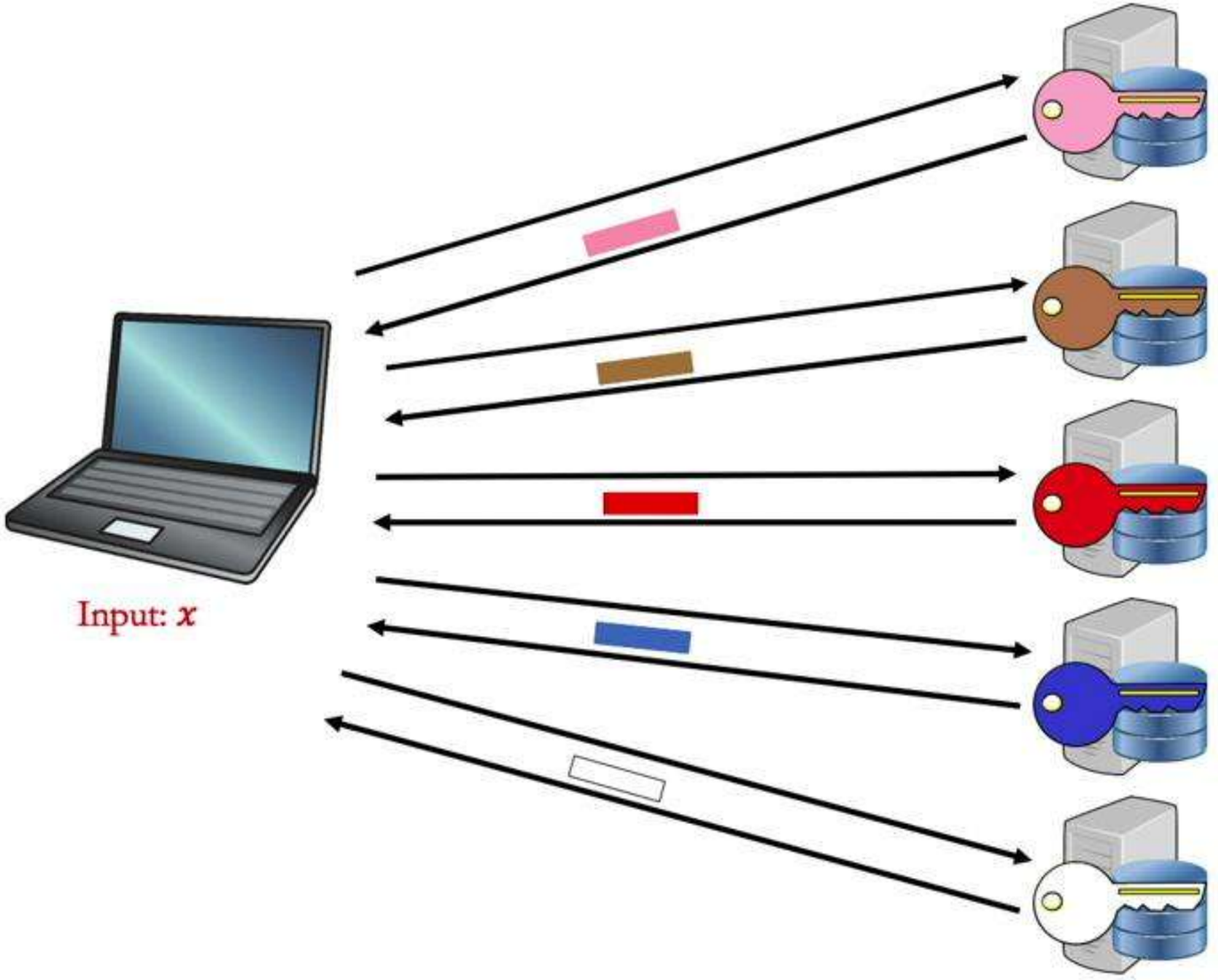
Secret key is secure!

Password is not secure (against offline dictionary attack)!

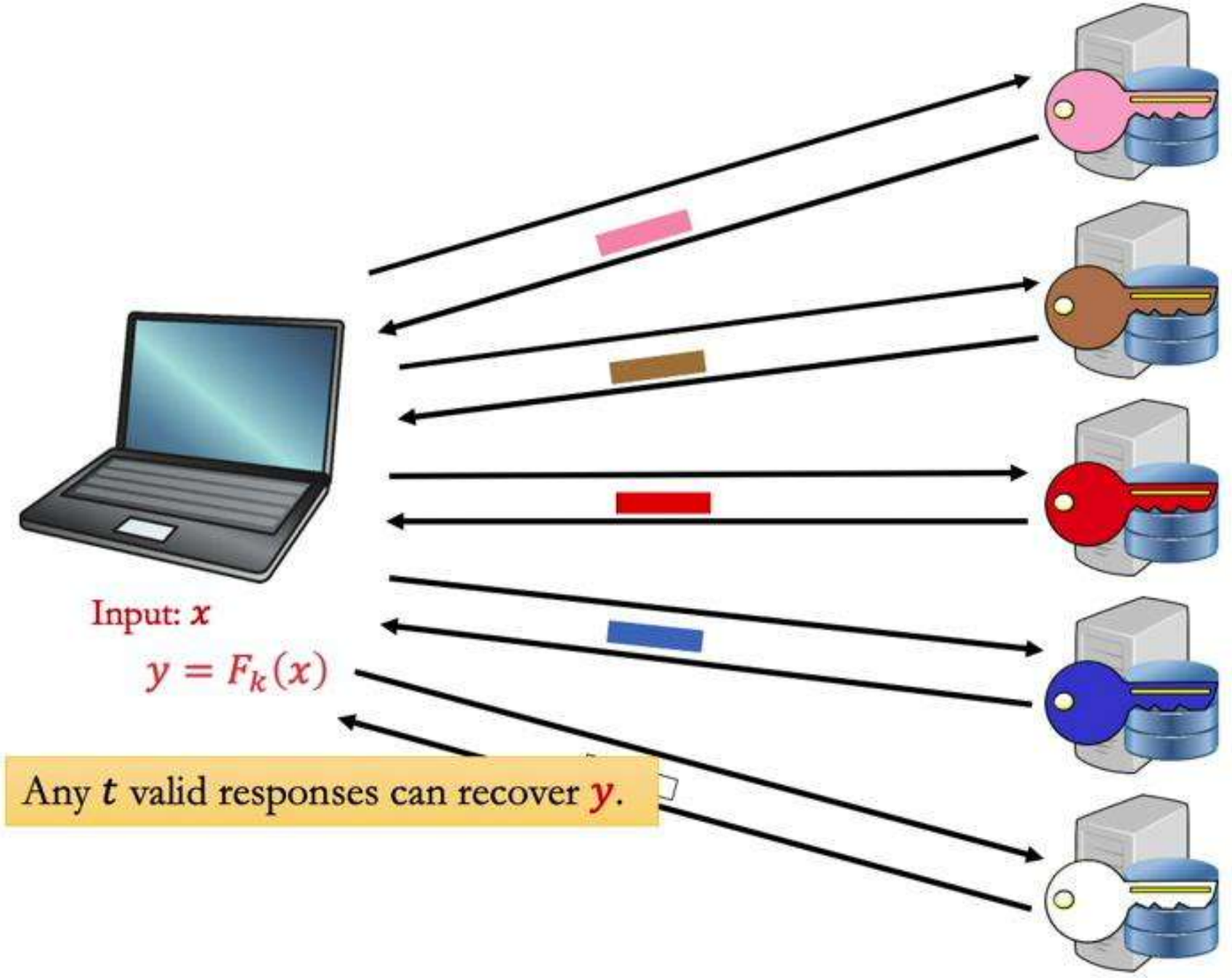
Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]

Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]

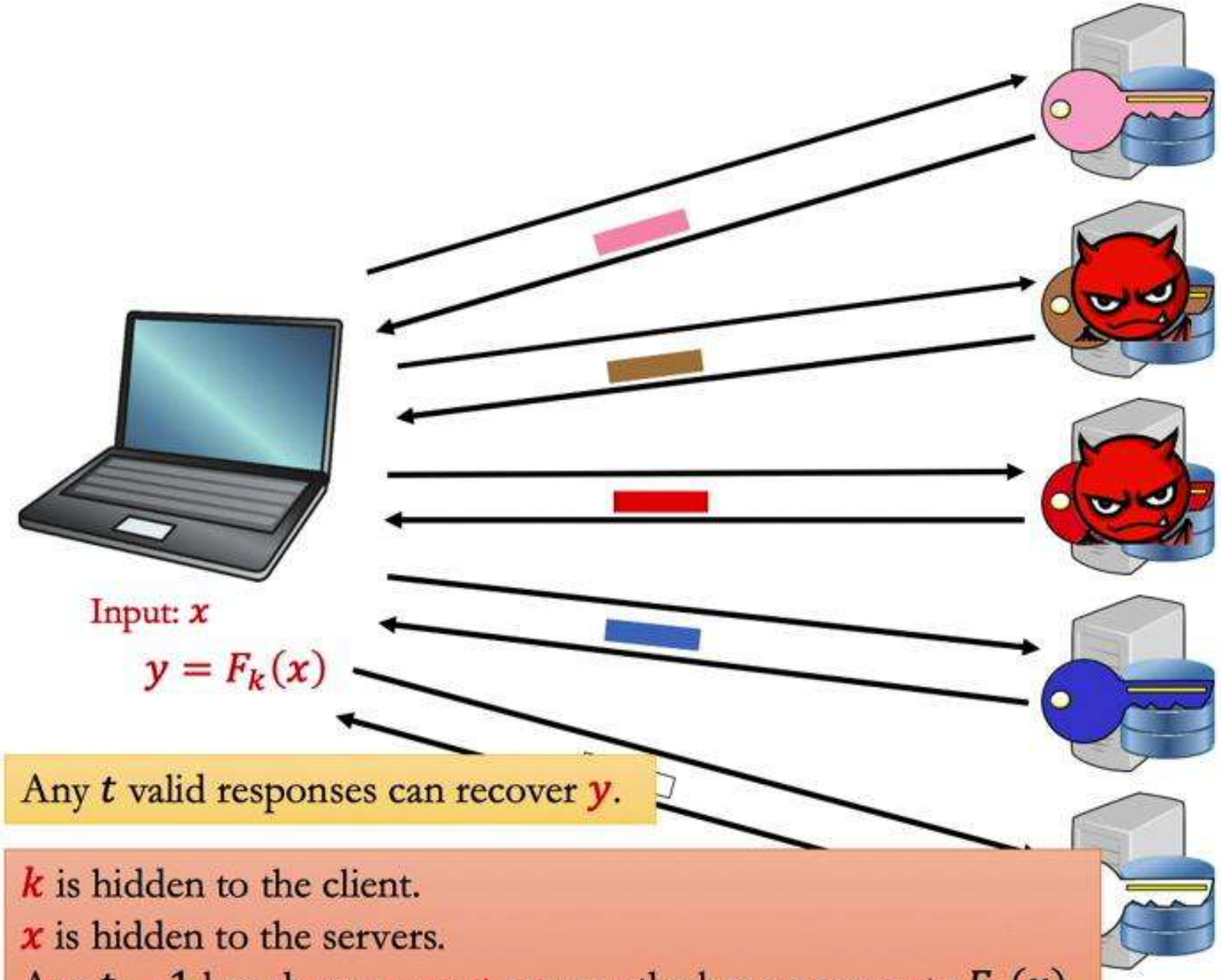
Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]



Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]



Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]



Any t valid responses can recover y.

k is hidden to the client.
x is hidden to the servers.
Any $t-1$ key shares cannot recover the key or compute $F_k(x)$.

Our Approach

- Non-Interactive Threshold Token Generation (NITTG)
  - Symmetric key based MAC [NPR'99, MPSWW'02]
  - Public key (DDH) based MAC [NPR'99]
  - RSA based digital signature [Shoup'00]
  - Pairing based digital signature [BLS'01, Boldyreva'03]

- Threshold Oblivious Pseudorandom Function (TOPRF) [JKKX'17]
  - Formalize game-based definition for our needs
  - Prove security for the construction

- Generic construction of PASTA from NITTG + TOPRF

Global Setup

Global Setup

Global Setup

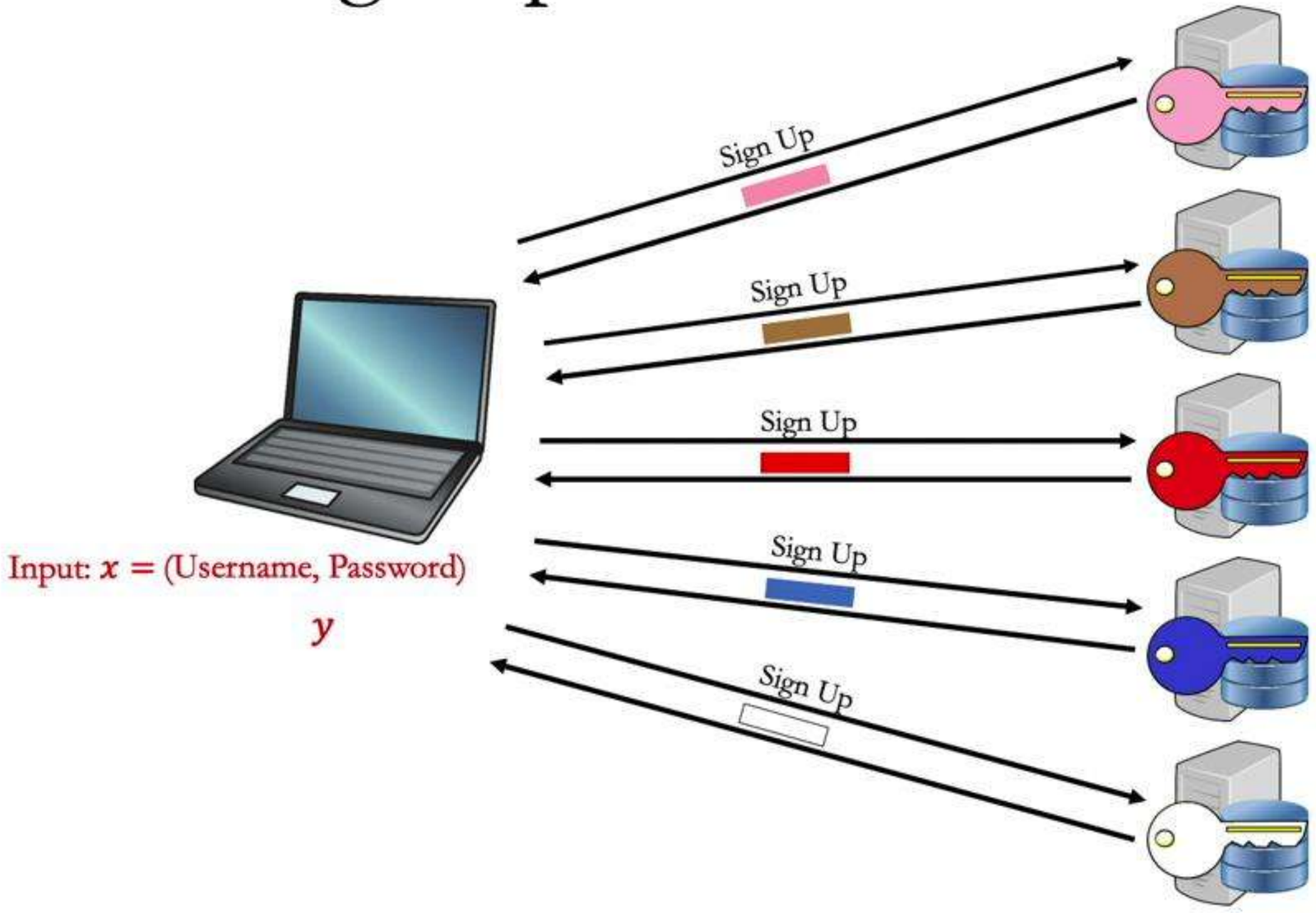
- 1 shared key for NITTG
- 1 shared key for TOPRF

User Sign Up

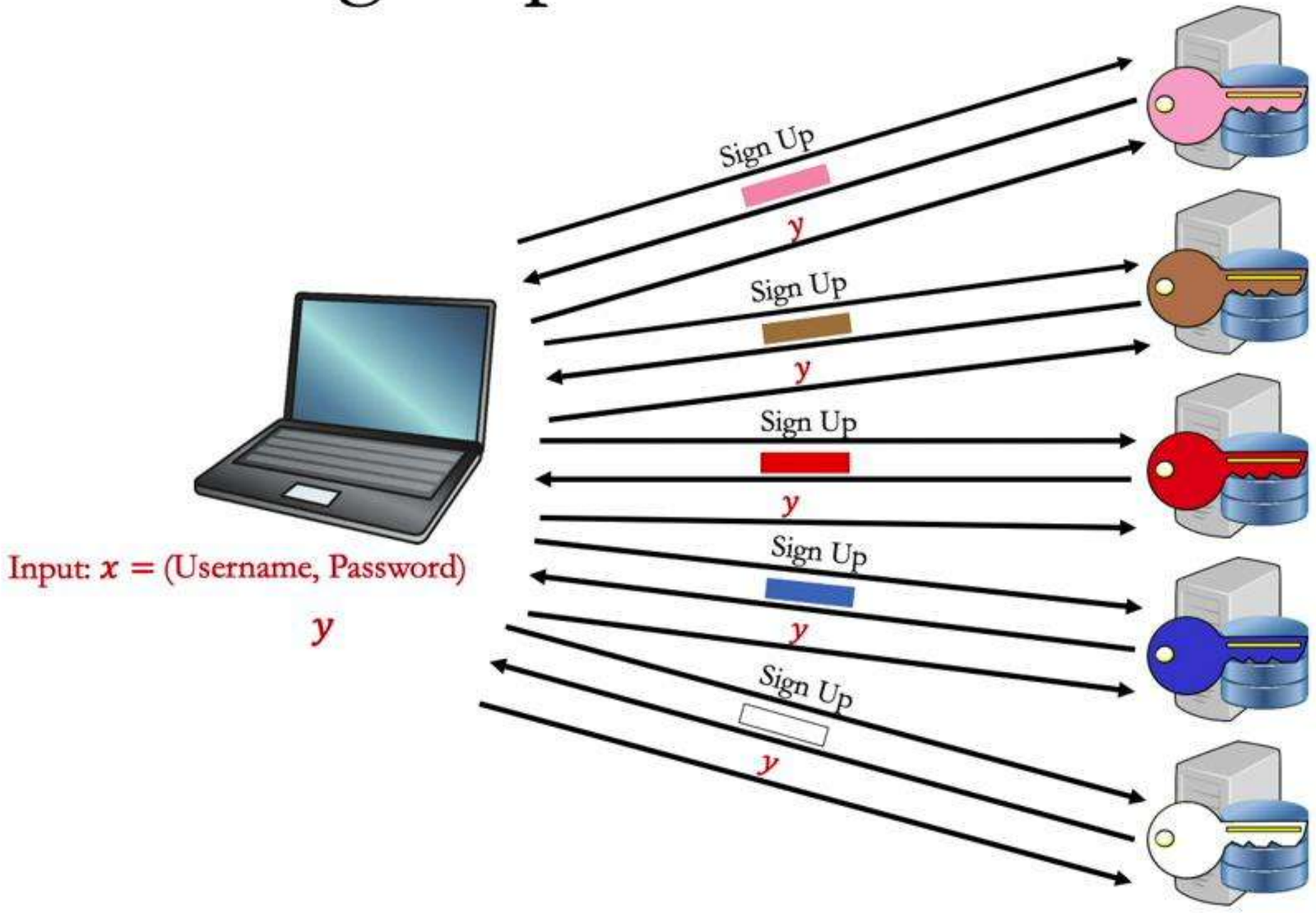
User Sign Up

(Username, Password)

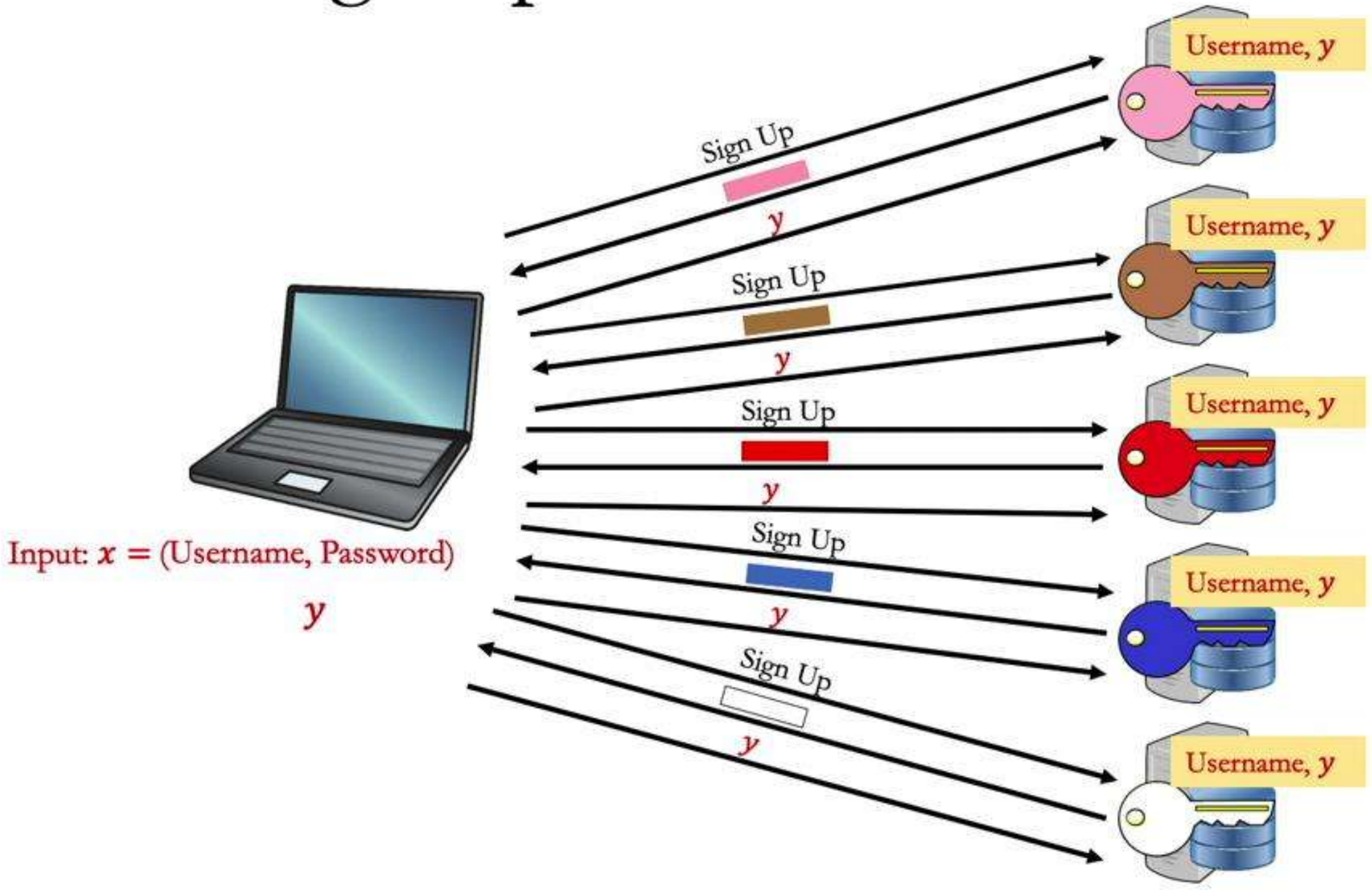
User Sign Up


Sign Up
Sign Up
Sign Up
Sign Up
Sign Up
Input: x = (Username, Password)
y

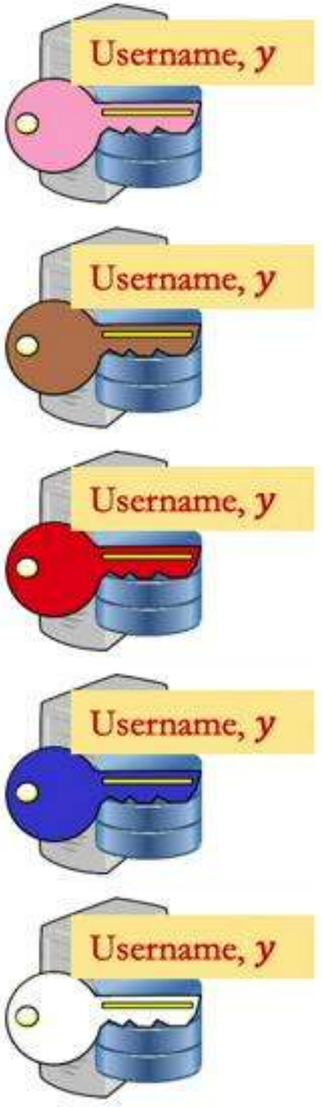
User Sign Up


Sign Up
y
Sign Up
y
Sign Up
y
Sign Up
y
Sign Up
y
Input: x = (Username, Password)
y

User Sign Up

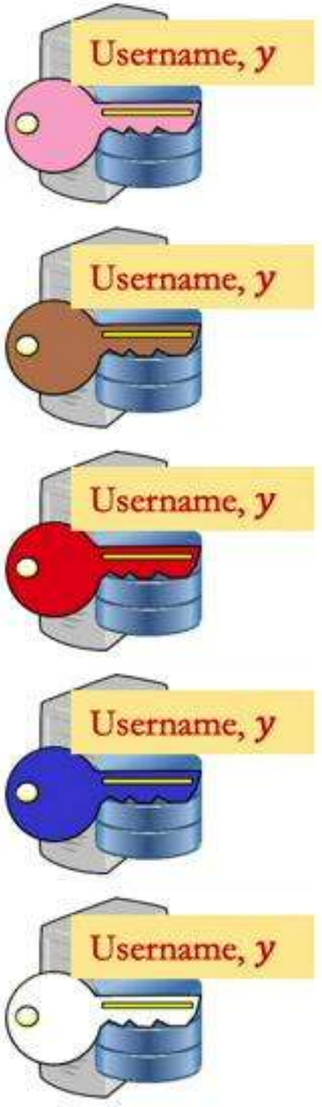

Username, y
Sign Up
y
Username, y
Sign Up
y
Username, y
Sign Up
y
Username, y
Sign Up
y
Username, y
Sign Up
y
Input: x = (Username, Password)
y

Request for Token

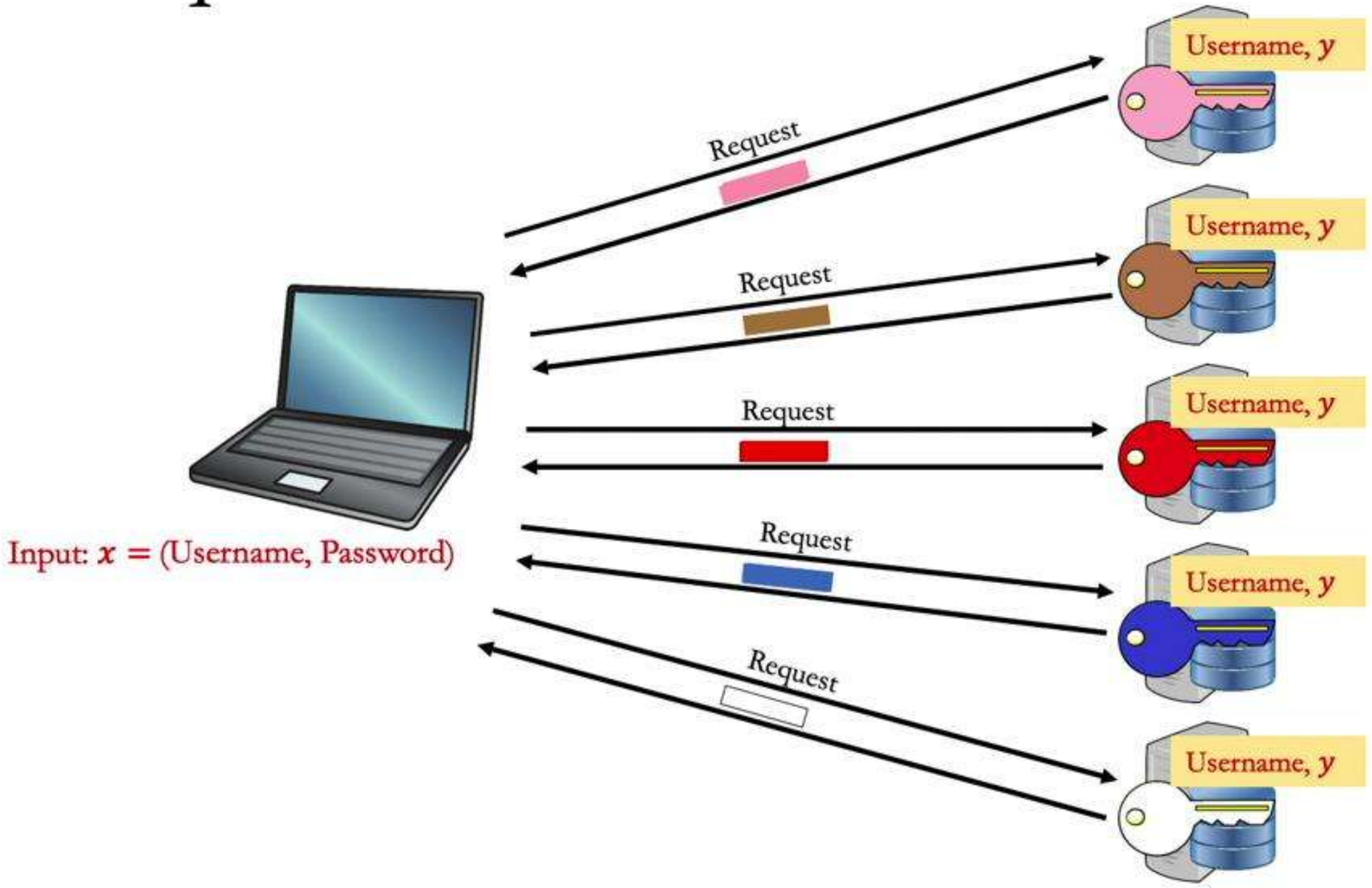


Request for Token

(Username, Password)



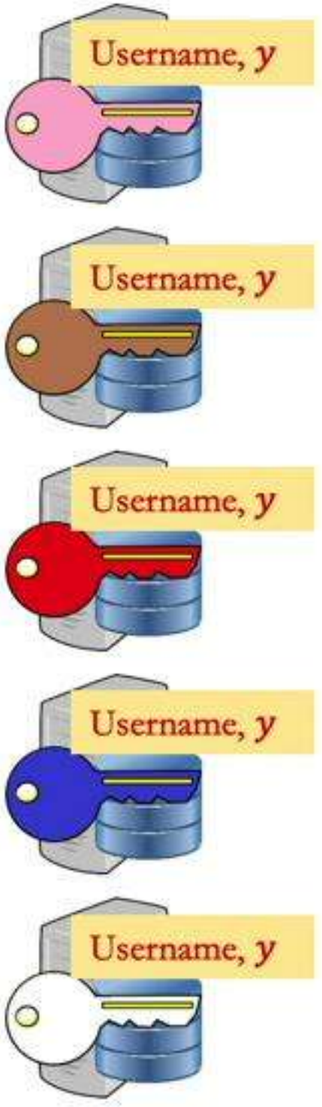
Request for Token



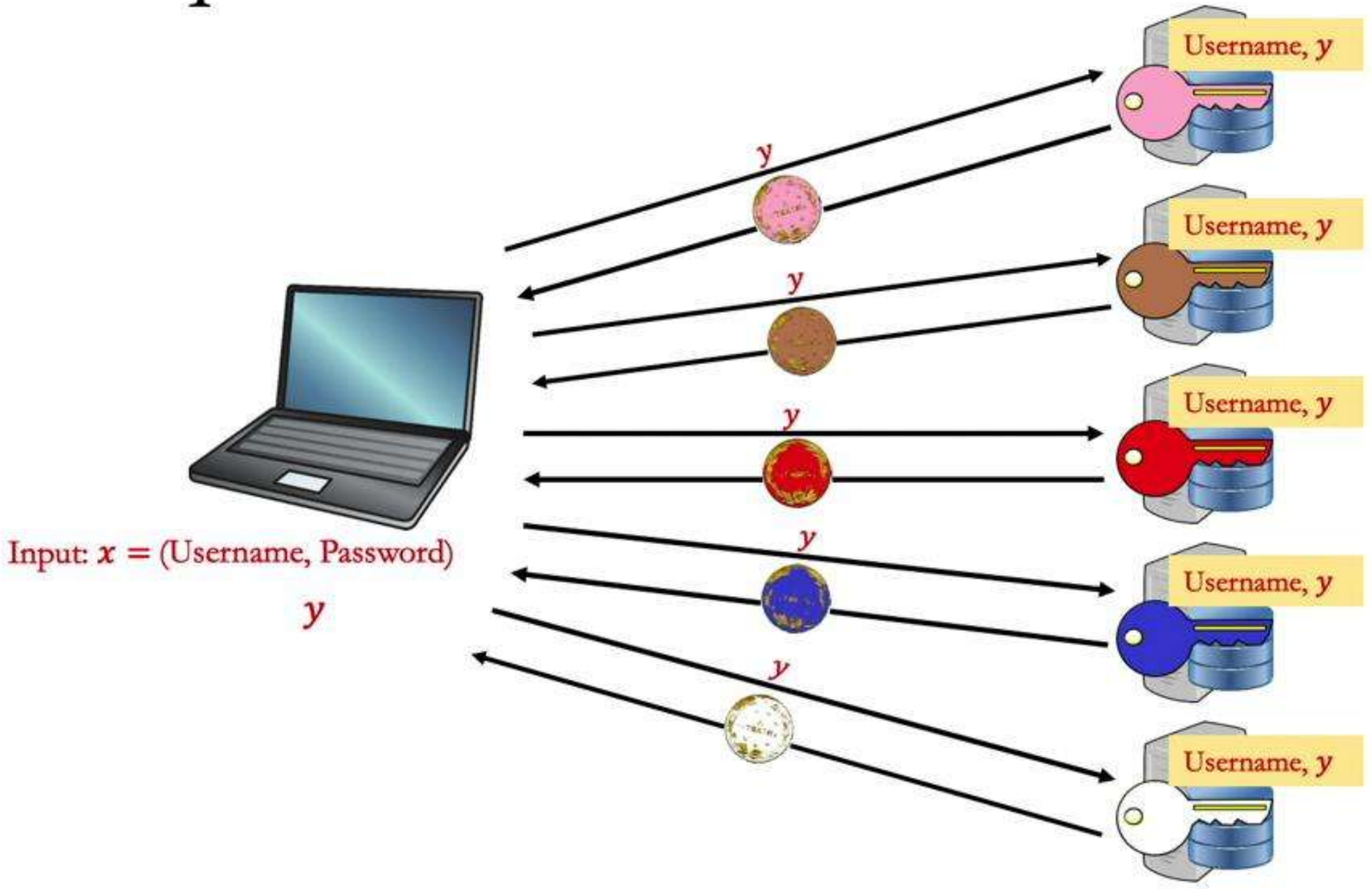
Request for Token – Cont'd

Input: x = (Username, Password)

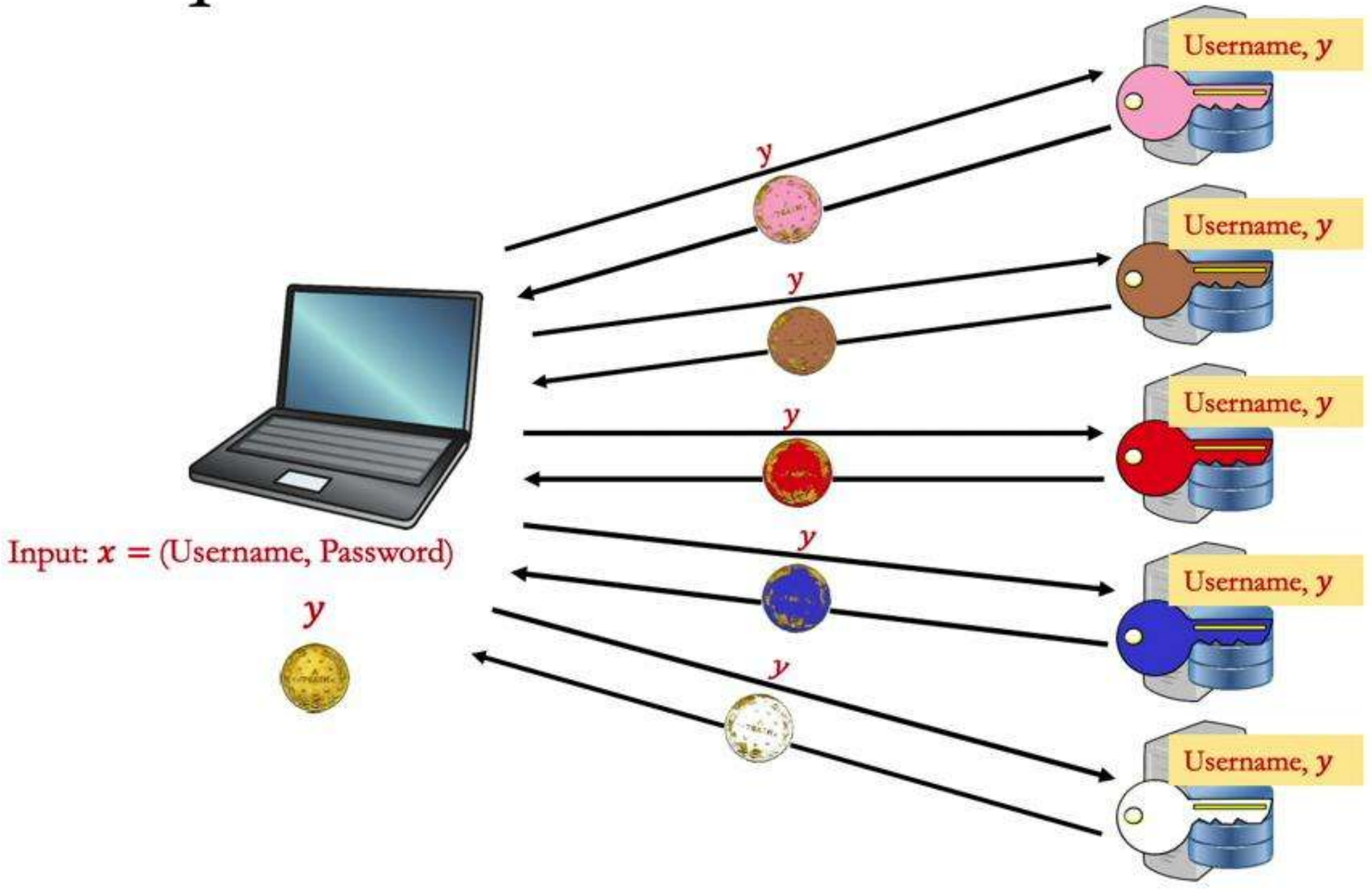
y



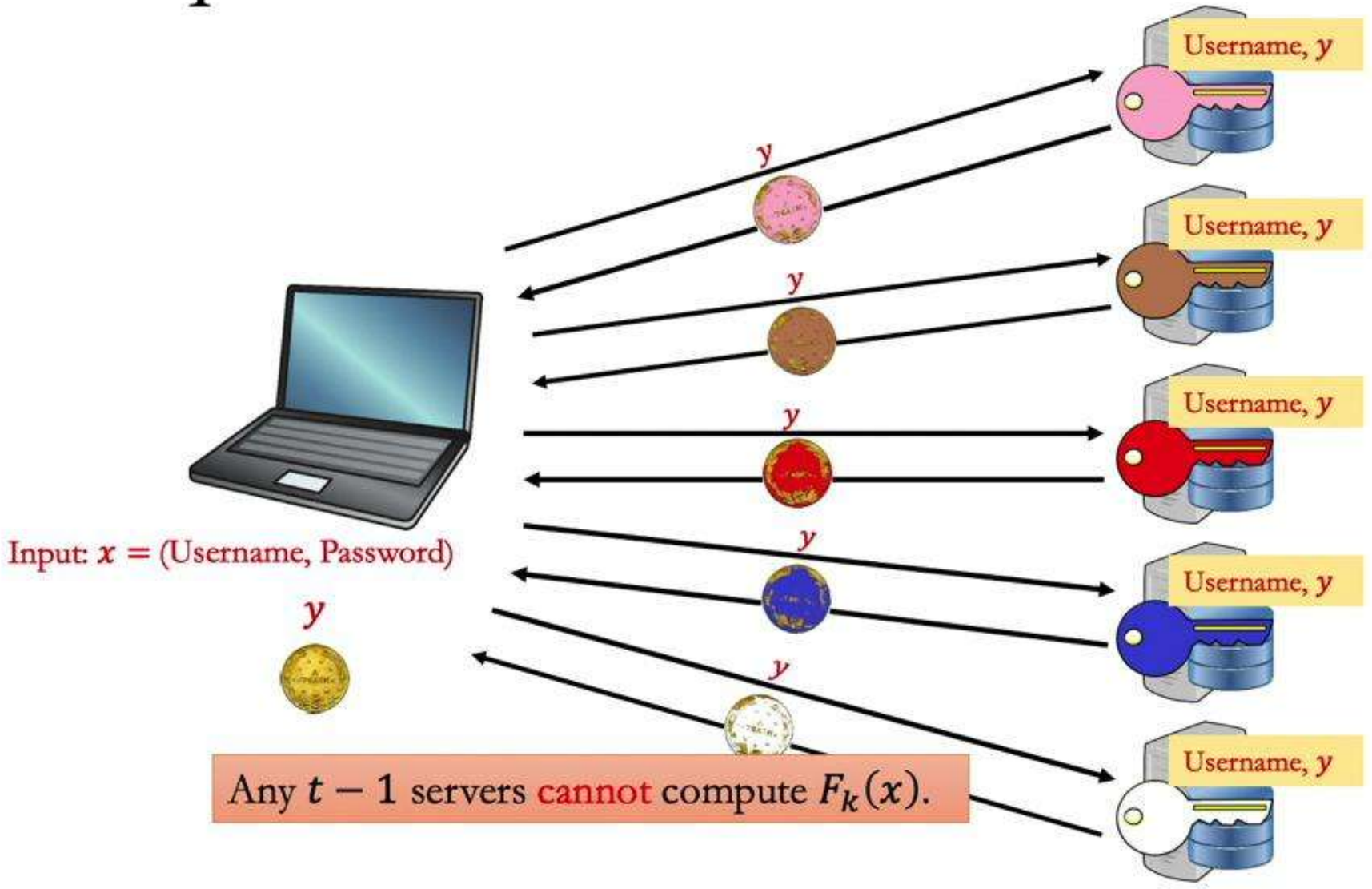
Request for Token – Cont'd



Request for Token – Cont'd

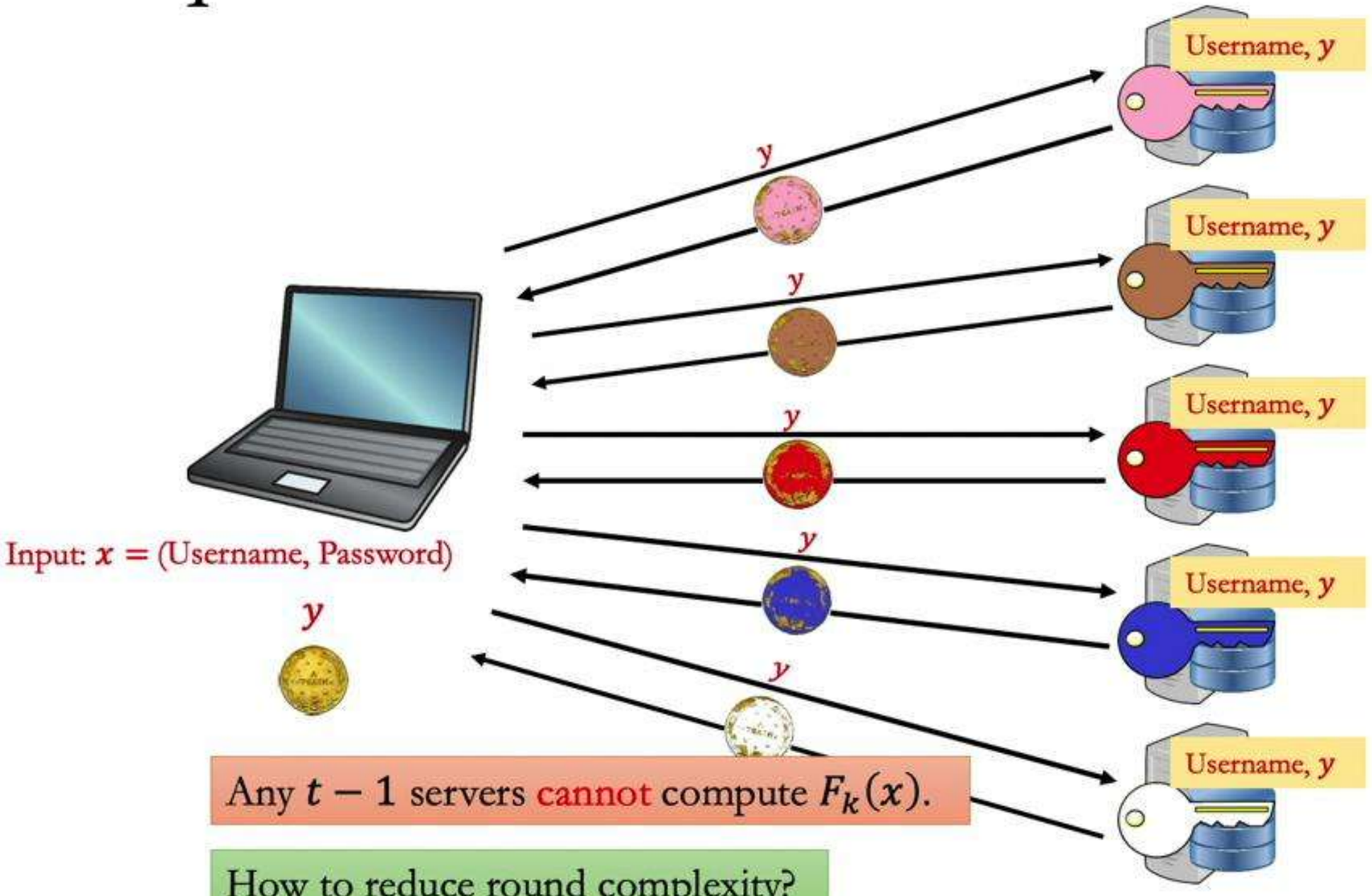


Request for Token – Cont'd



Any $t-1$ servers cannot compute $F_k(x)$.

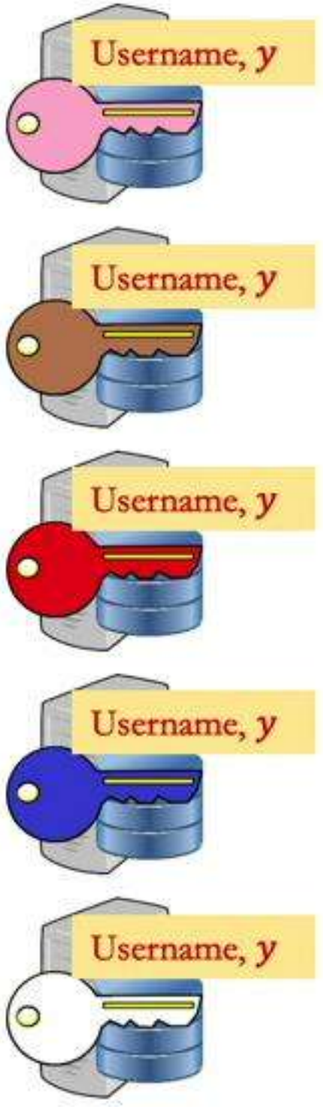
Request for Token – Cont'd



Any $t-1$ servers cannot compute $F_k(x)$.

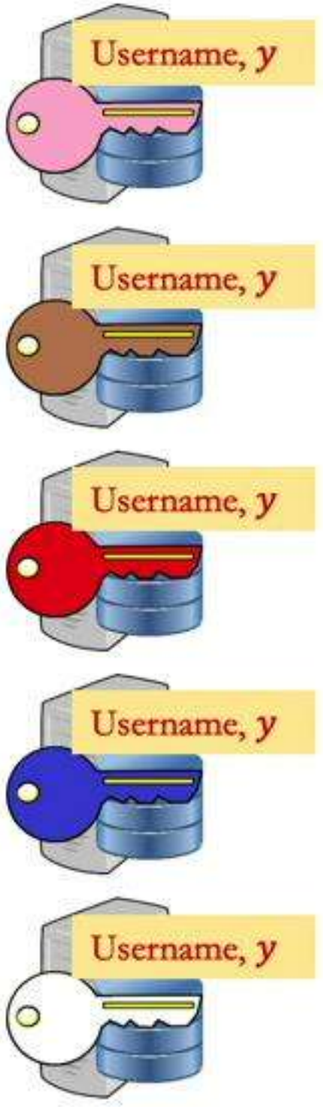
How to reduce round complexity?

Request for Token – Minimal Interaction

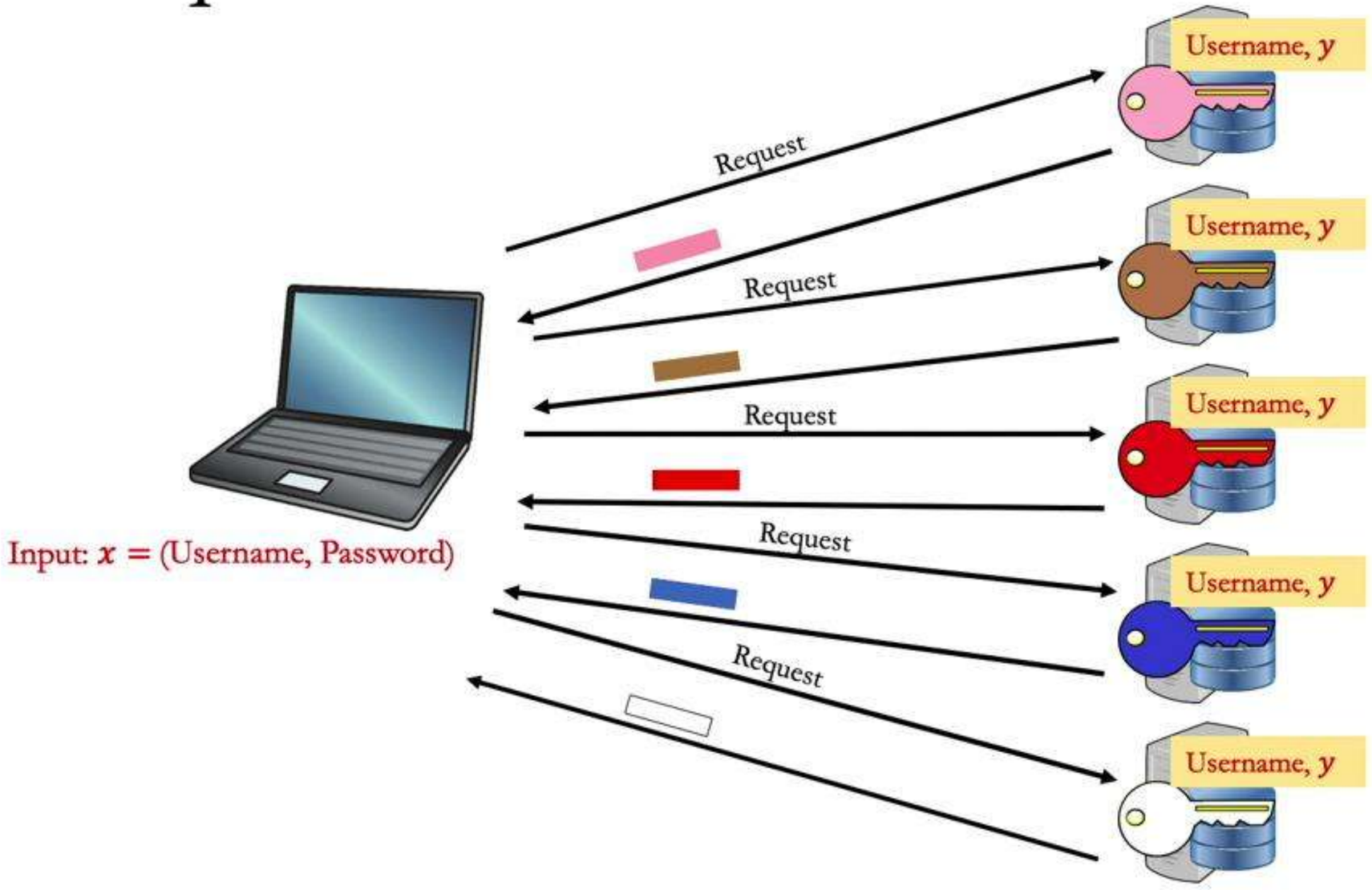


Request for Token – Minimal Interaction

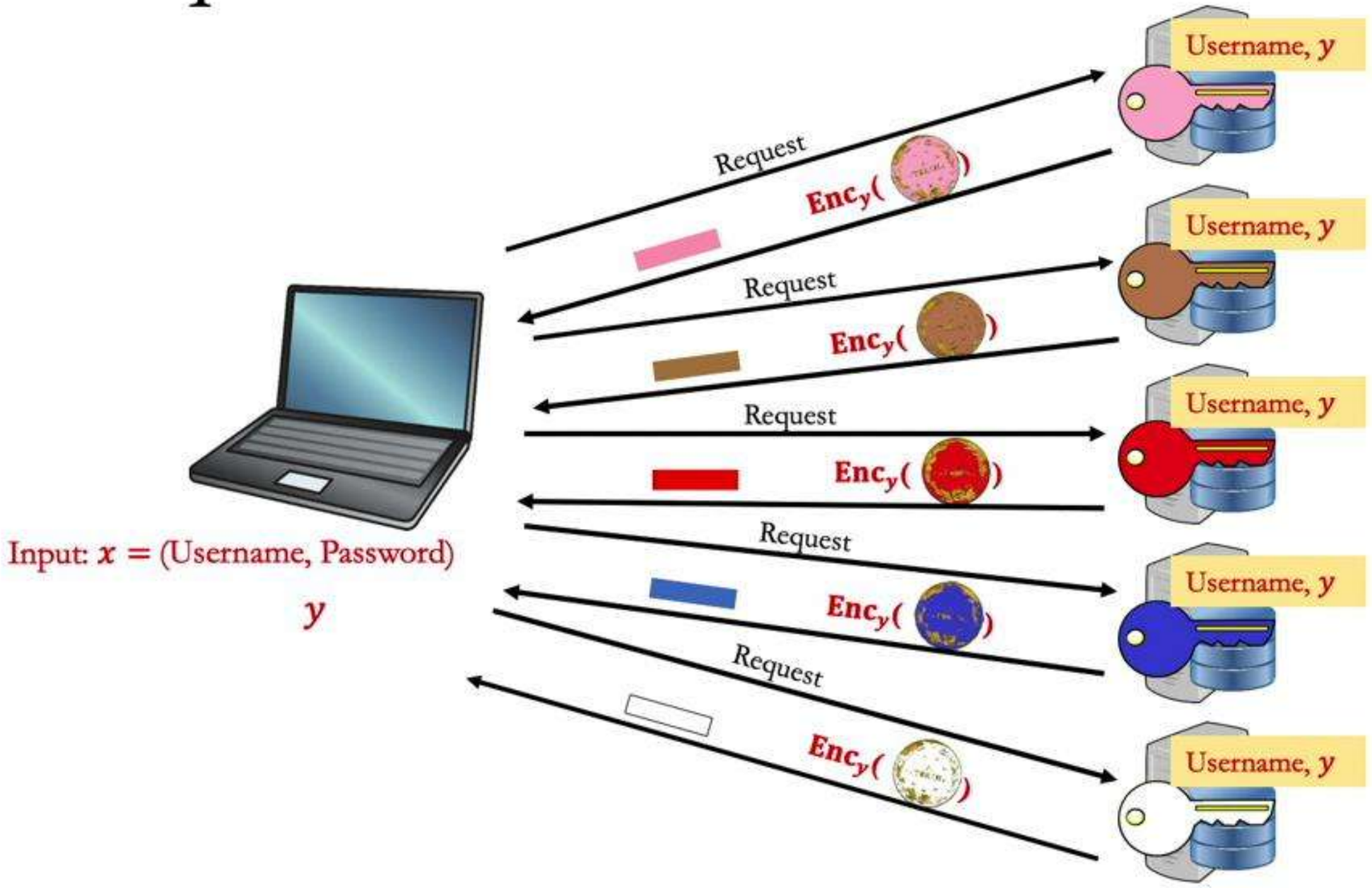
(Username, Password)



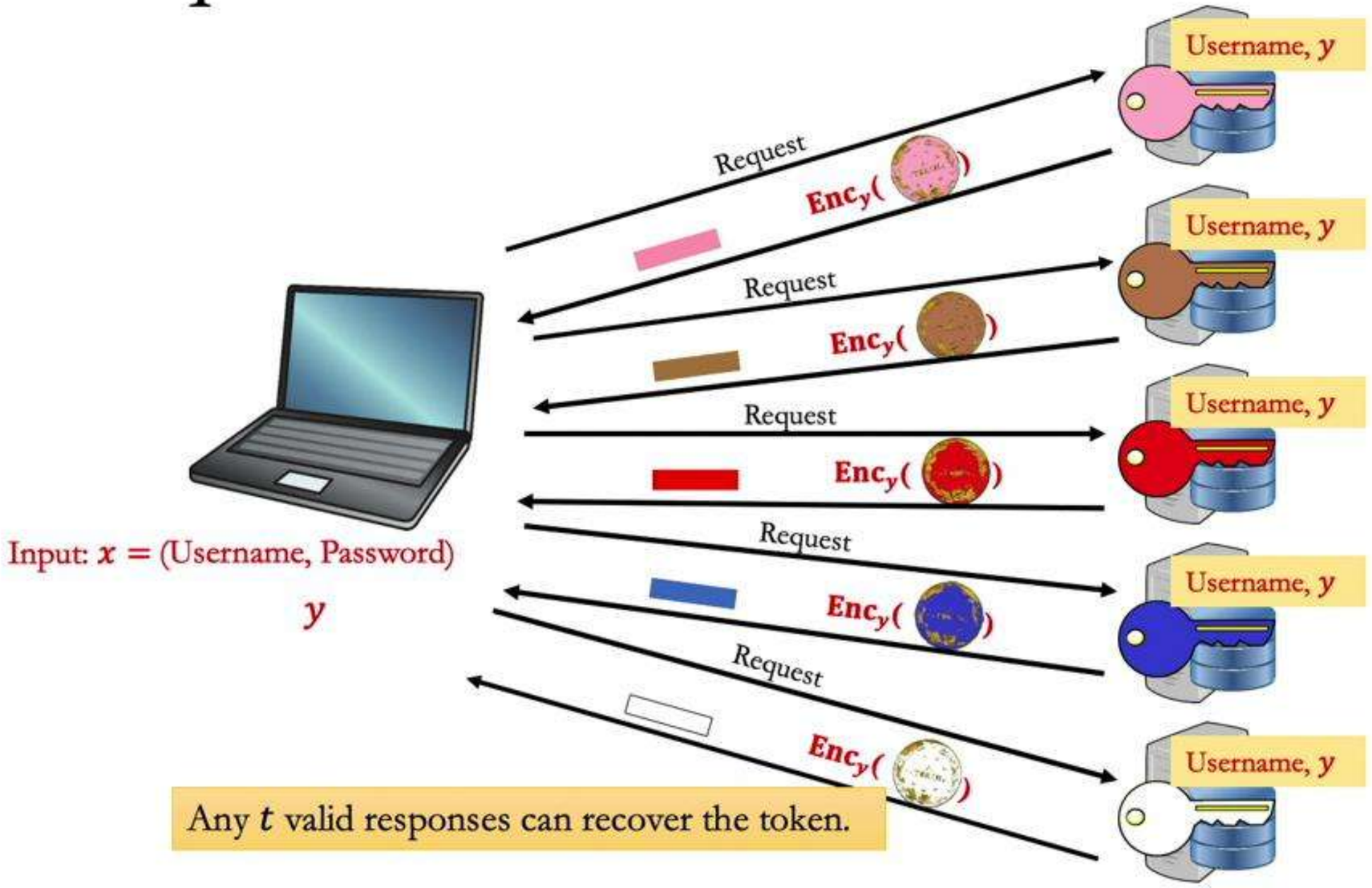
Request for Token – Minimal Interaction



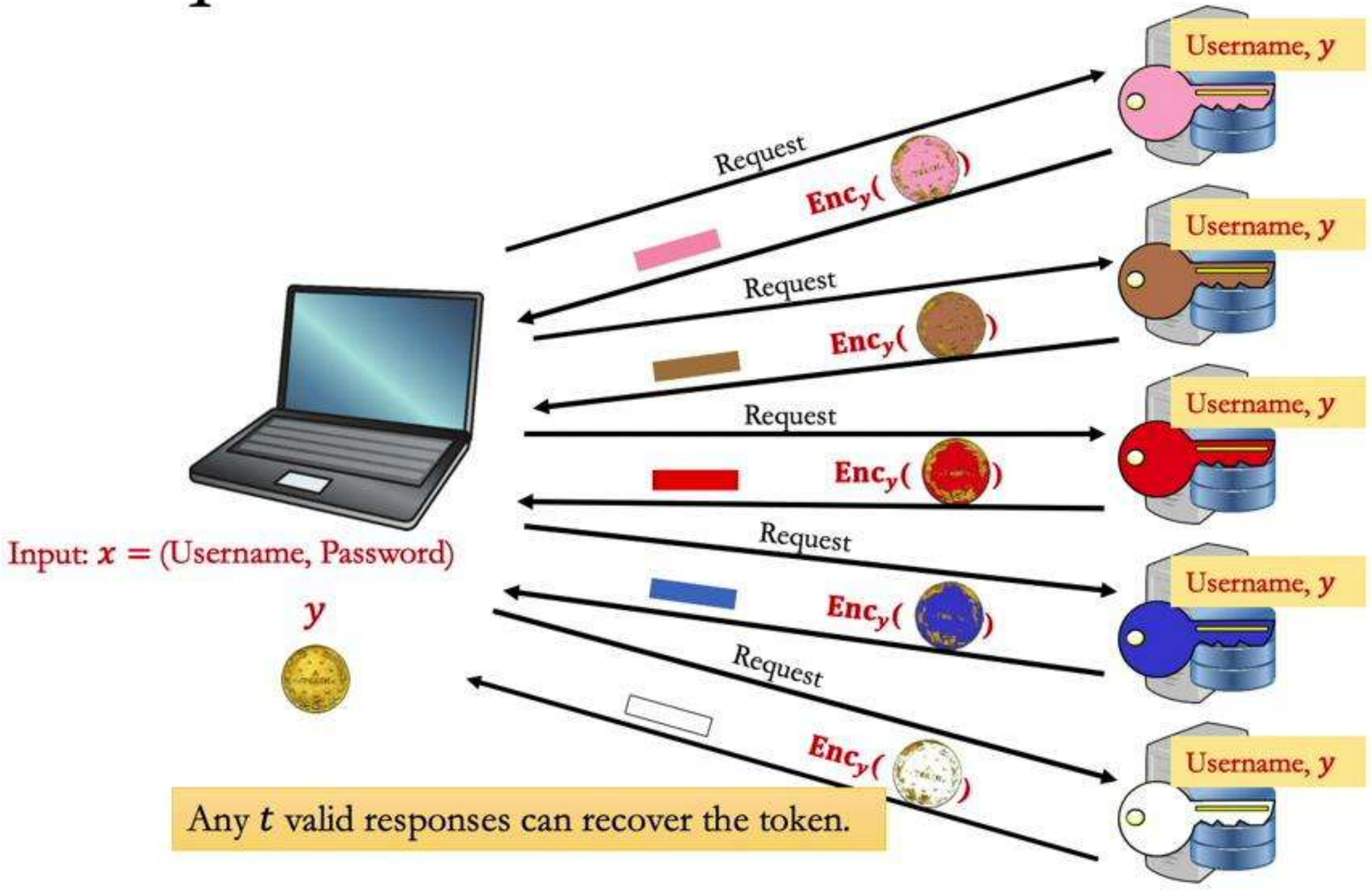
Request for Token – Minimal Interaction



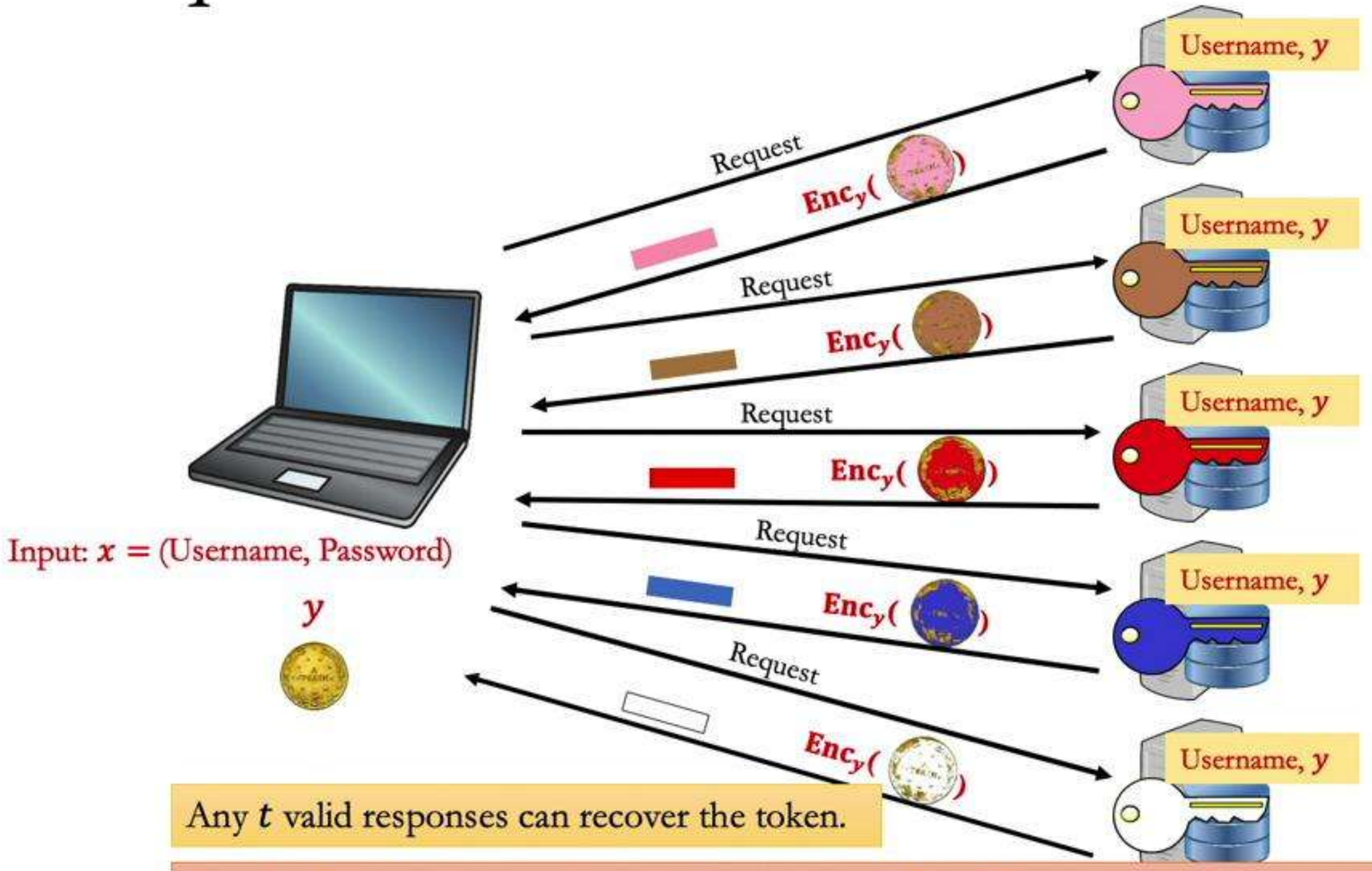
Request for Token – Minimal Interaction



Request for Token – Minimal Interaction



Request for Token – Minimal Interaction



Any t valid responses can recover the token.

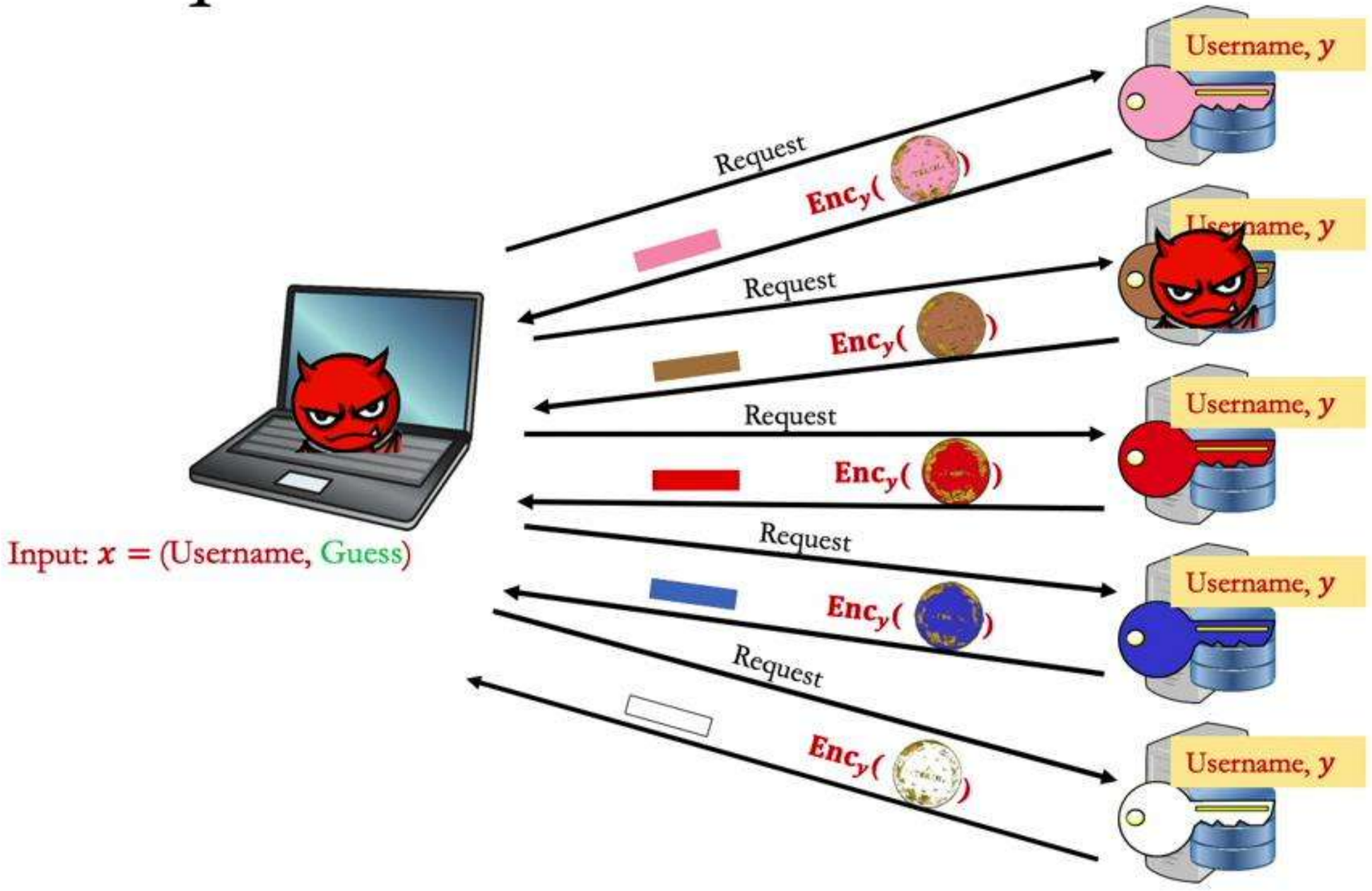
Any $t-1$ servers cannot recover the password (by offline attack) or forge a token?

Request for Token

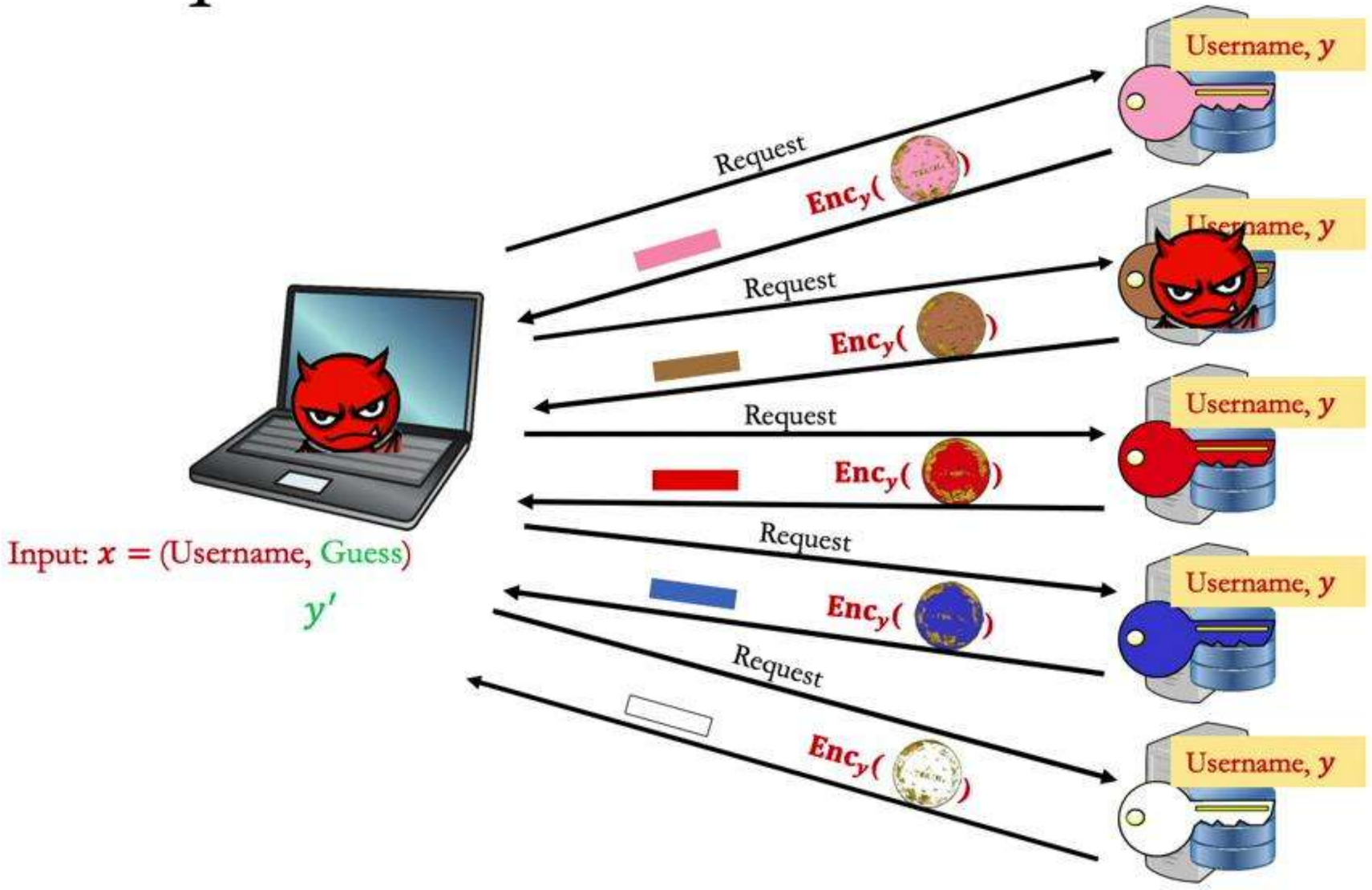
Request for Token

(Username, Guess)

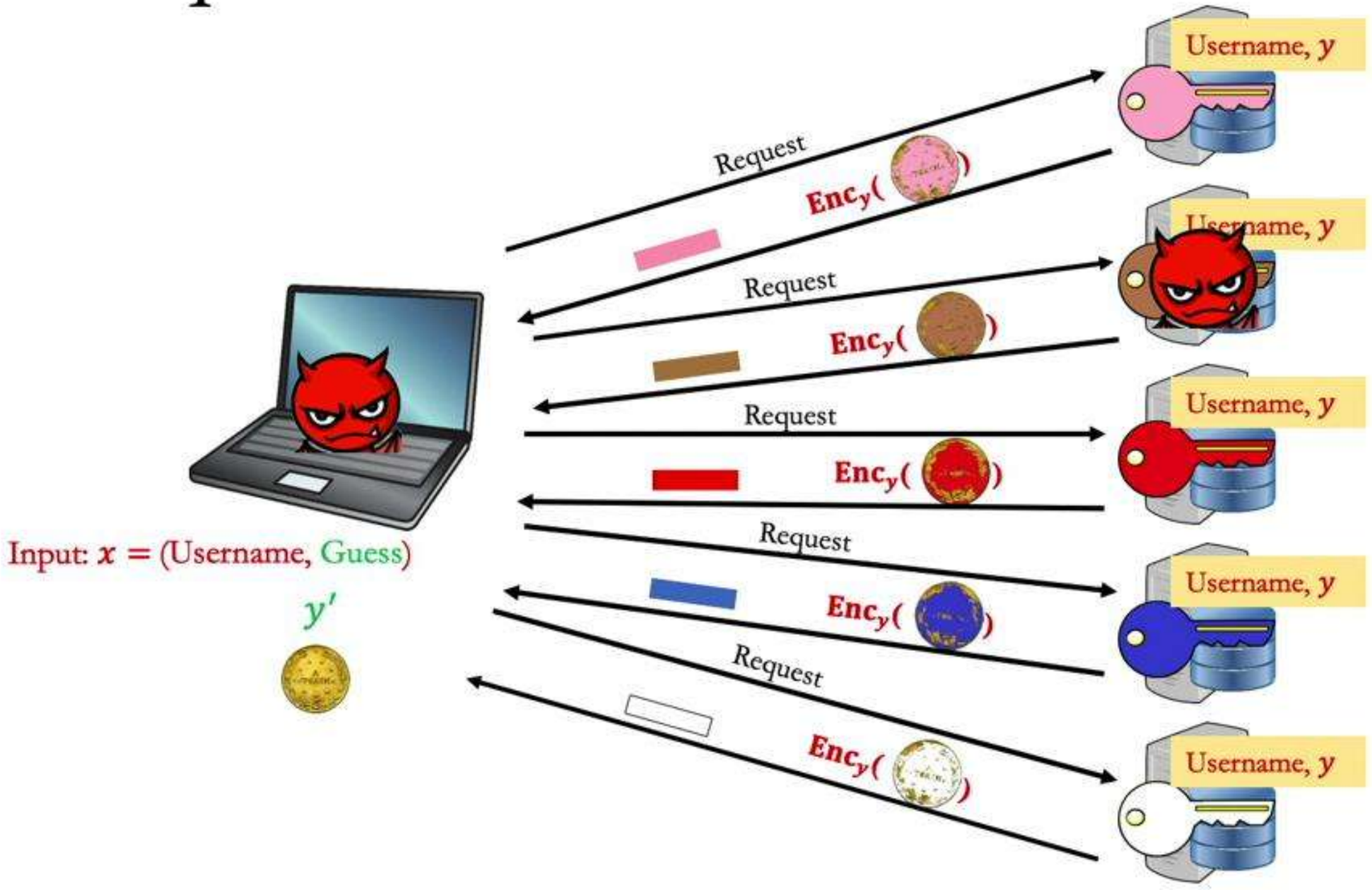
Request for Token



Request for Token



Request for Token

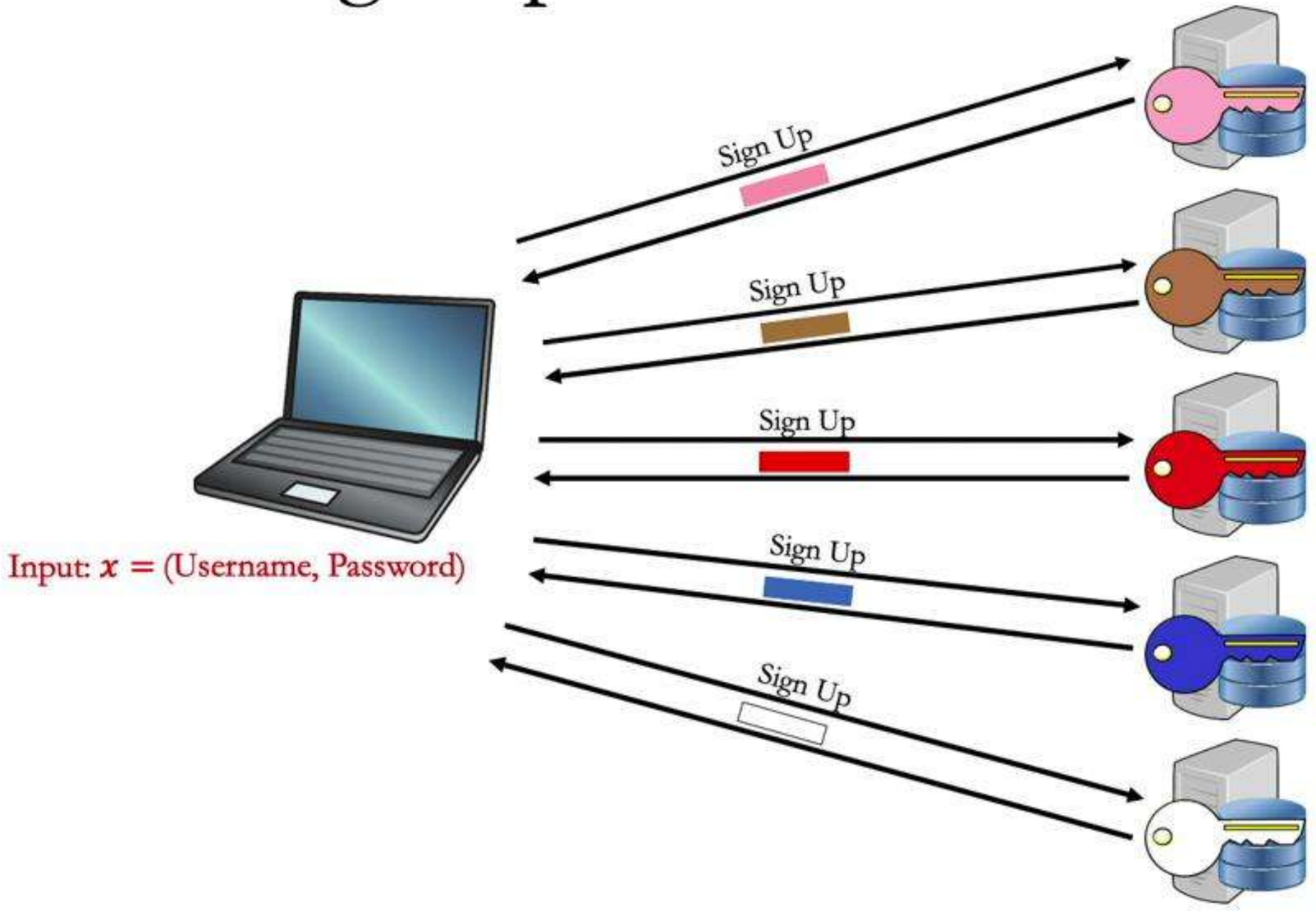


User Sign Up – A Fix

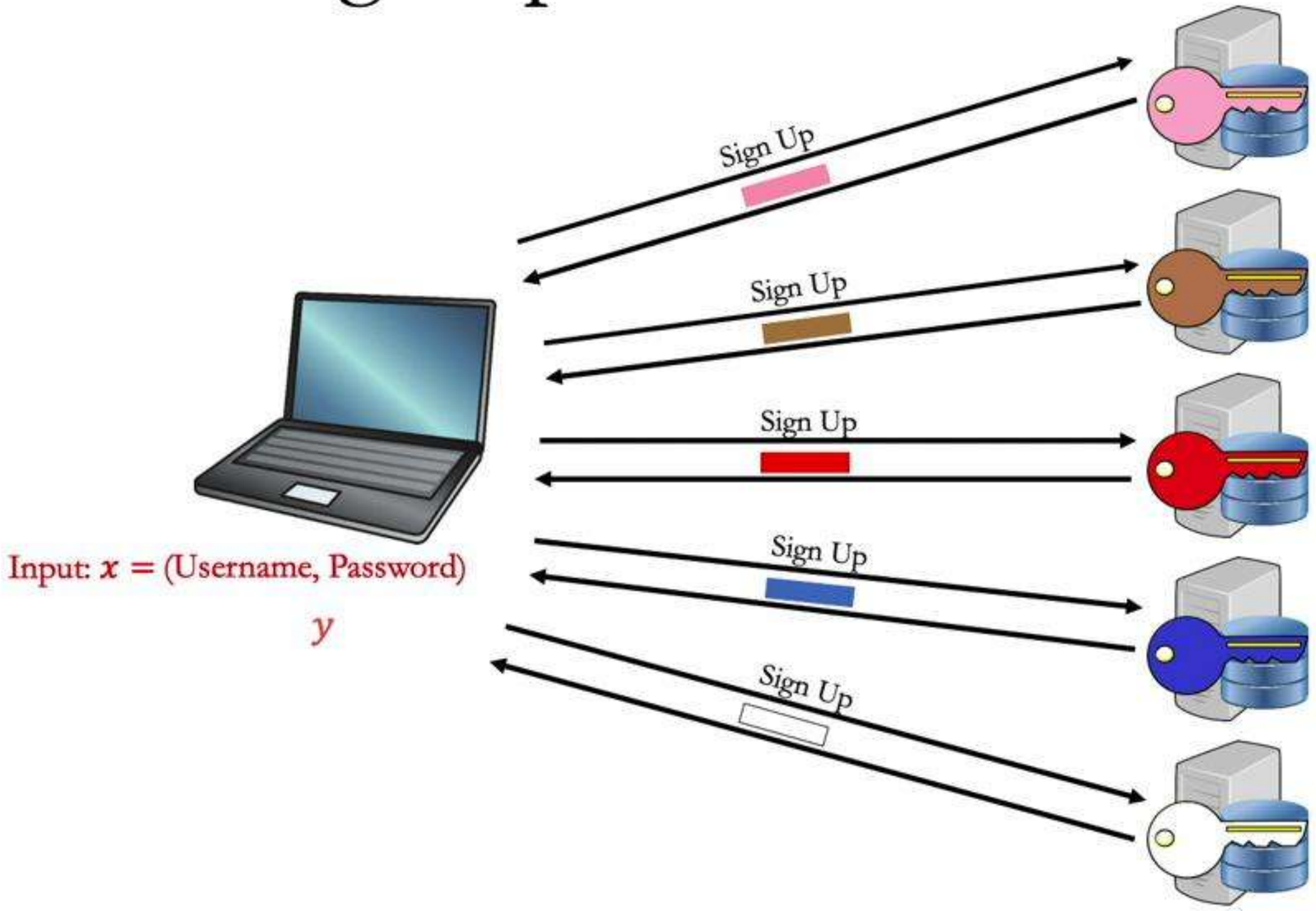
User Sign Up – A Fix

(Username, Password)

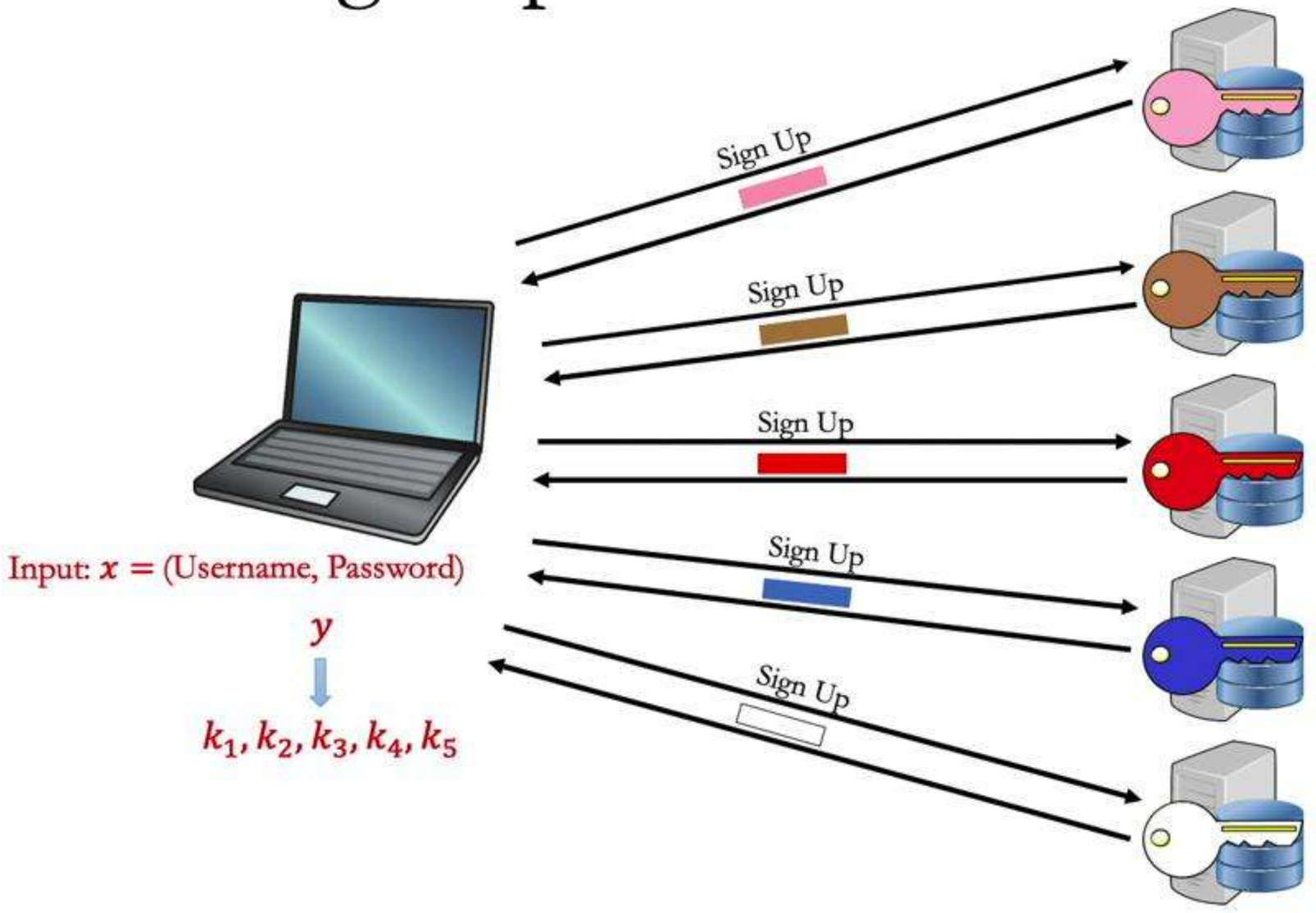
User Sign Up – A Fix



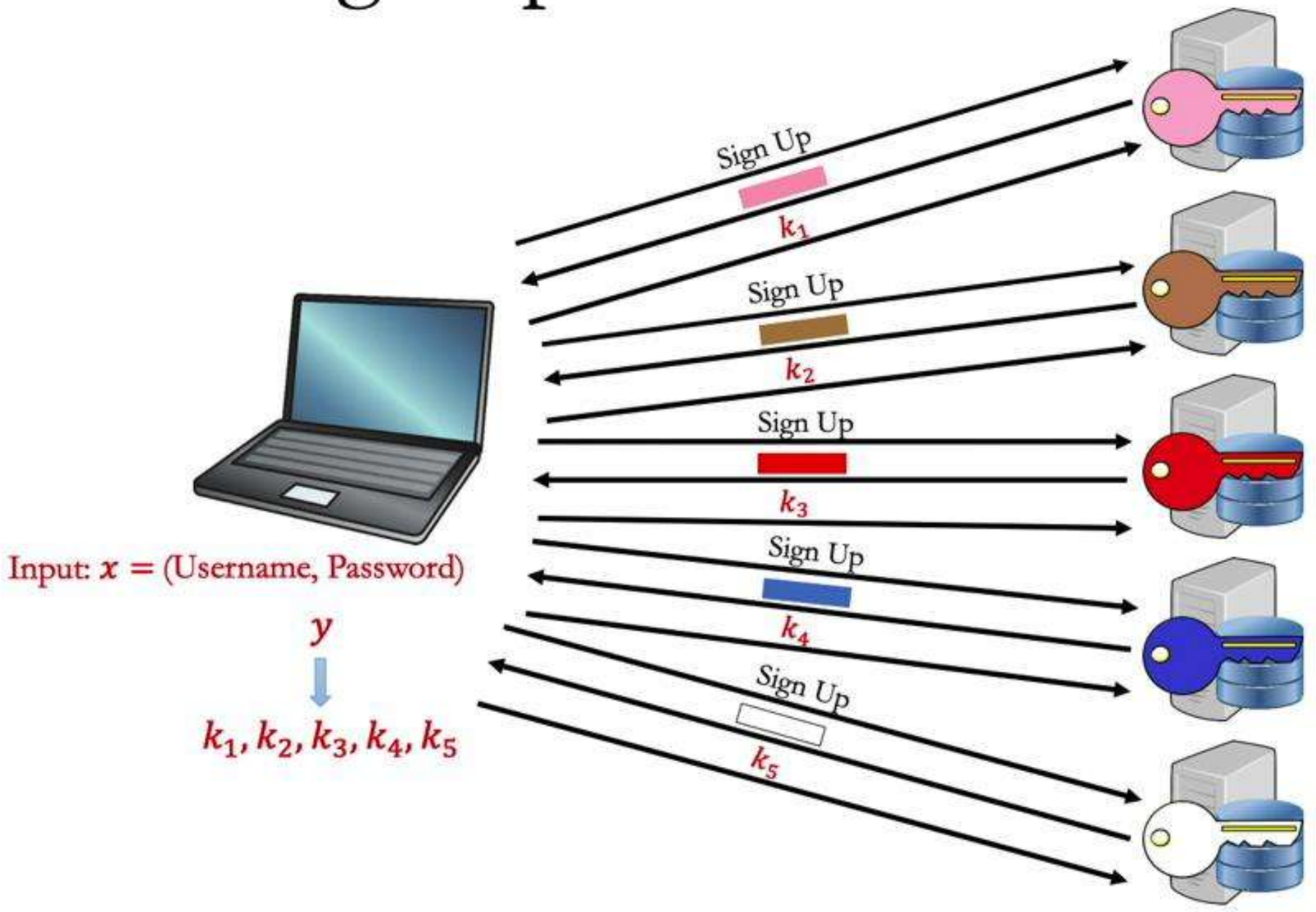
User Sign Up – A Fix



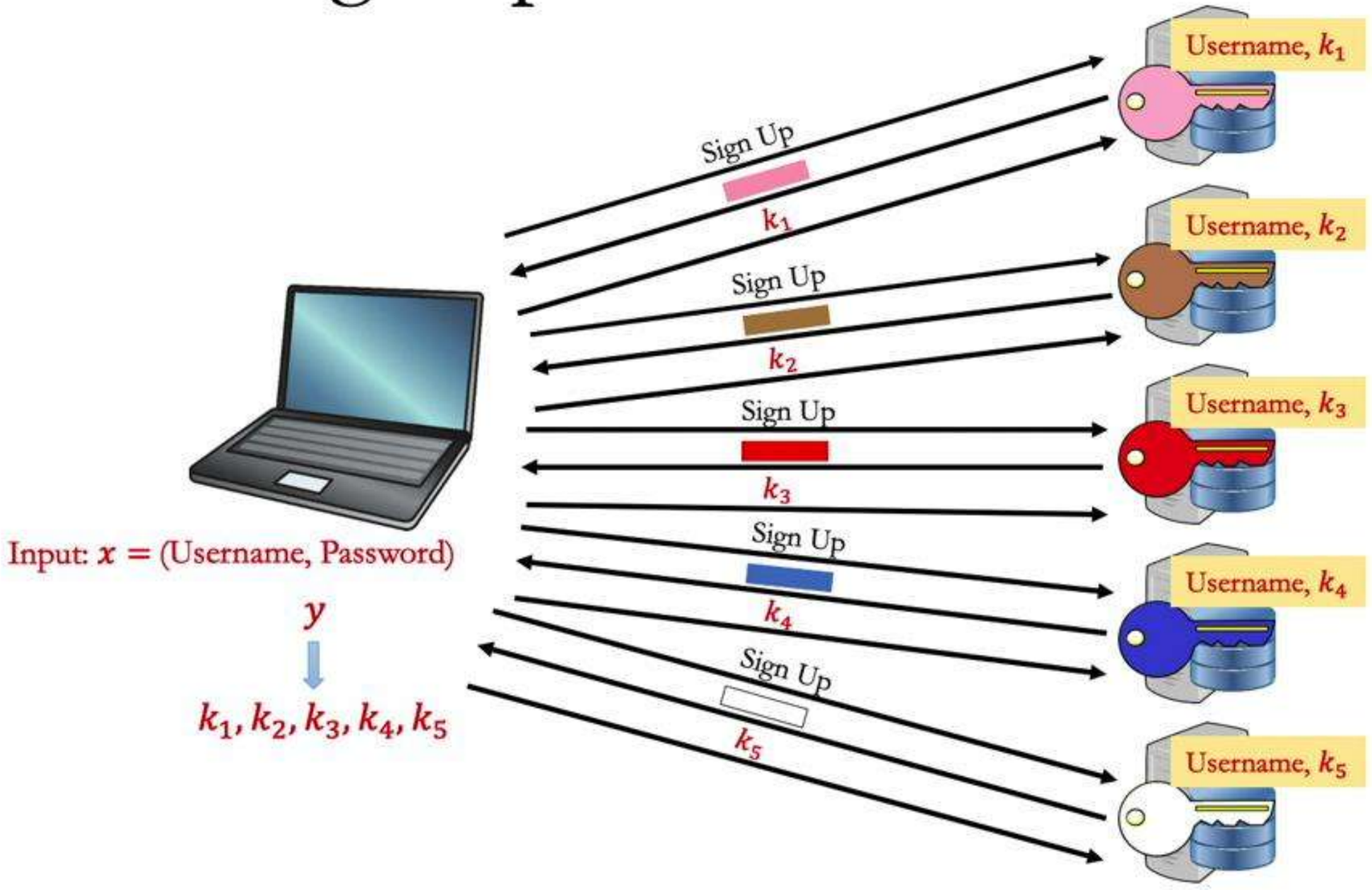
User Sign Up – A Fix



User Sign Up – A Fix

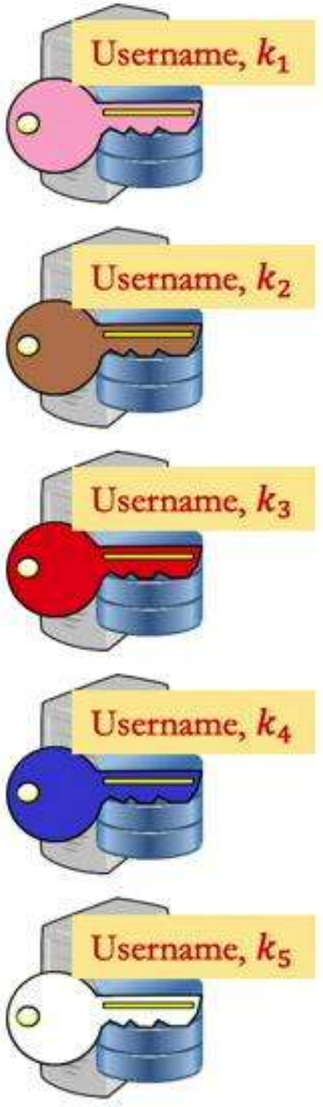


User Sign Up – A Fix

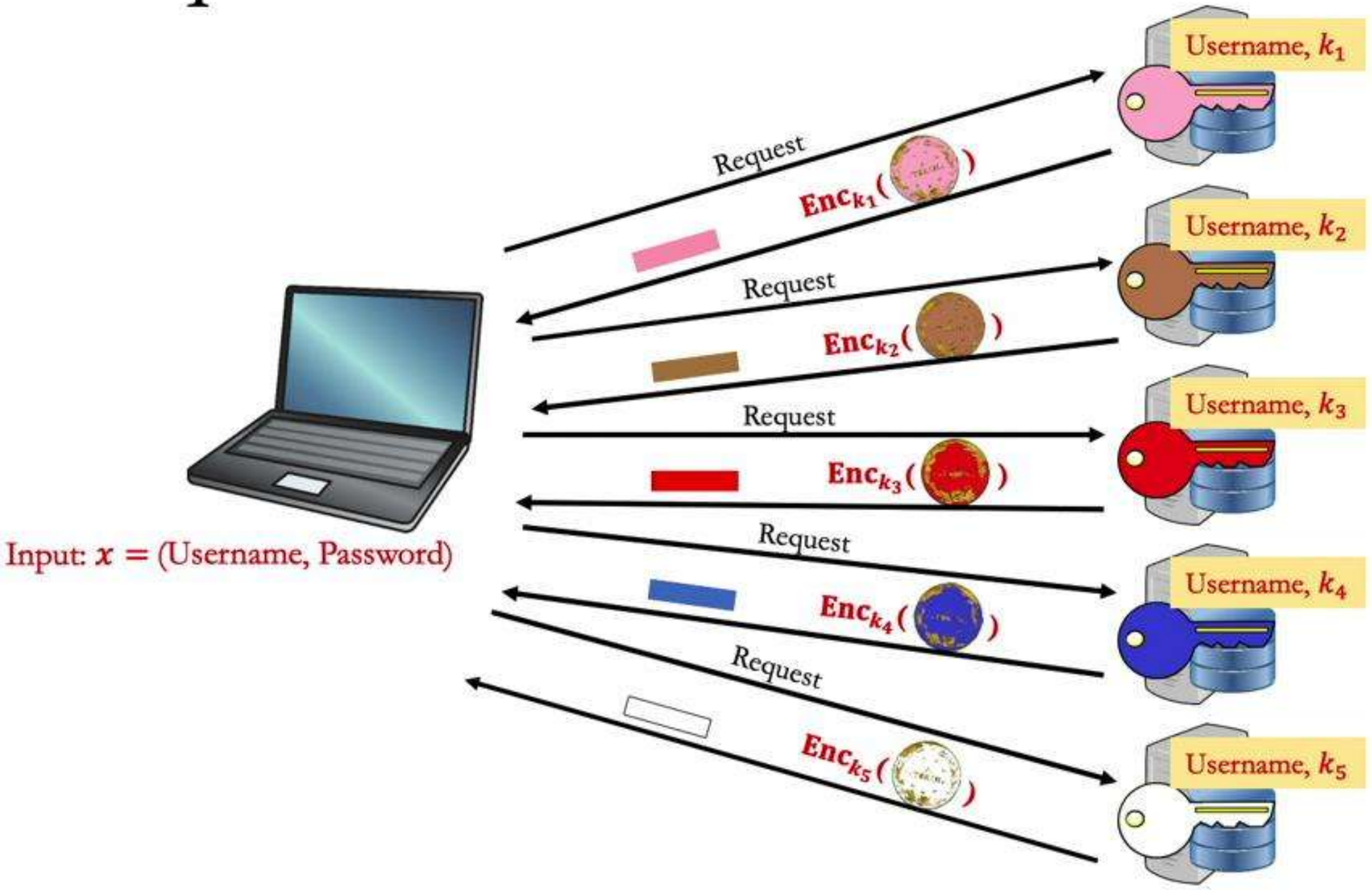


Request for Token – A Fix

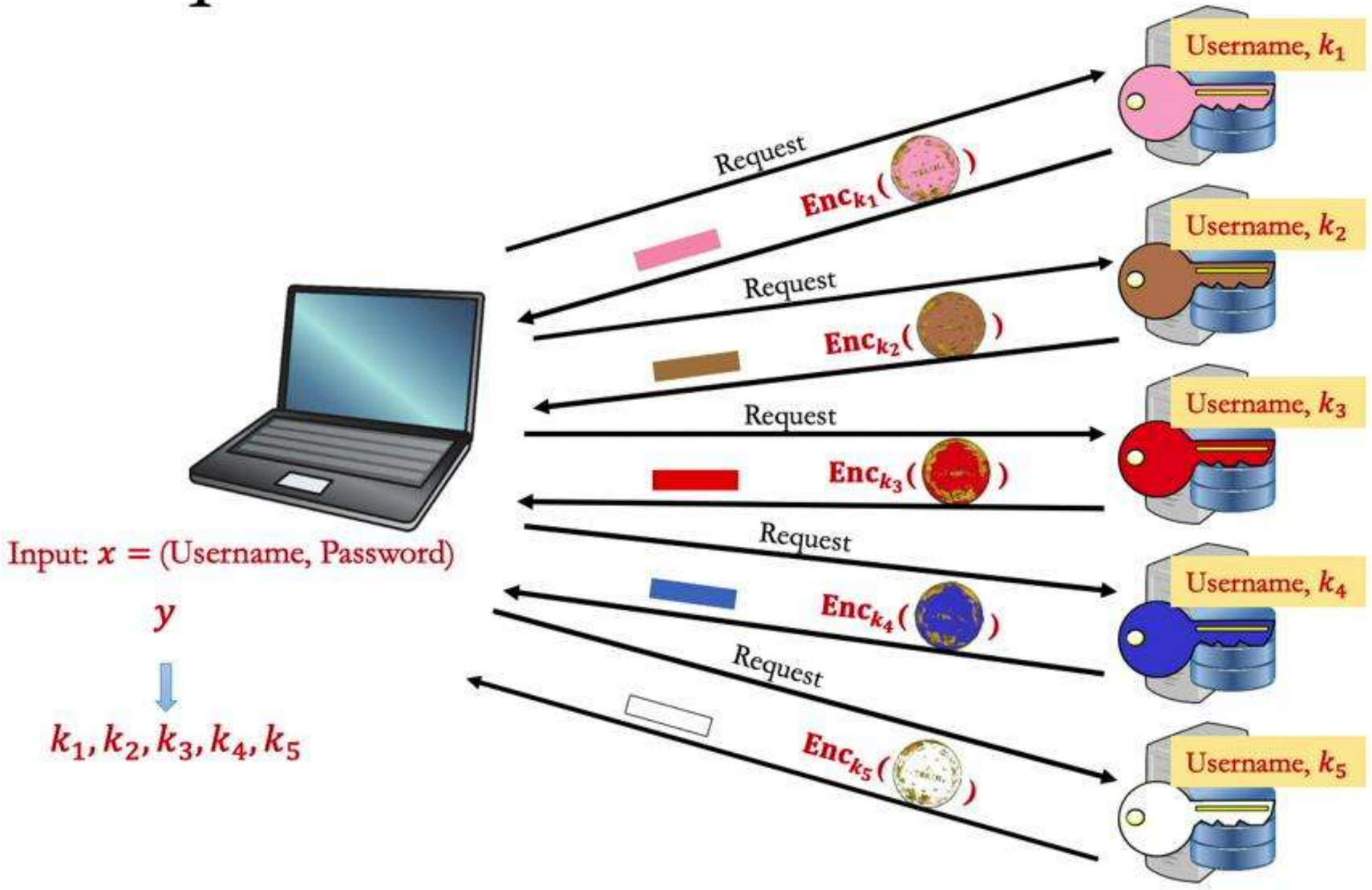
(Username, Password)



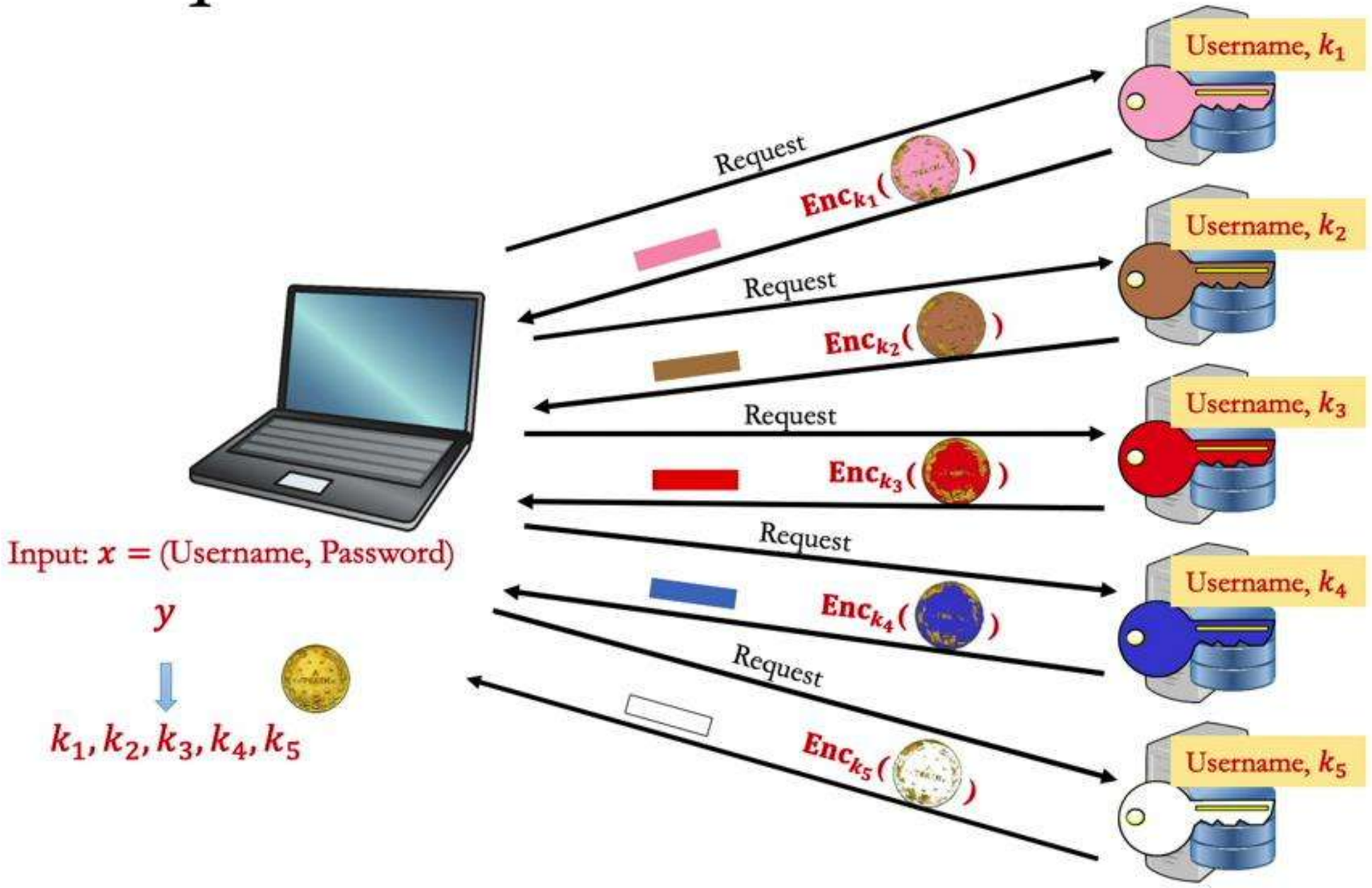
Request for Token – A Fix



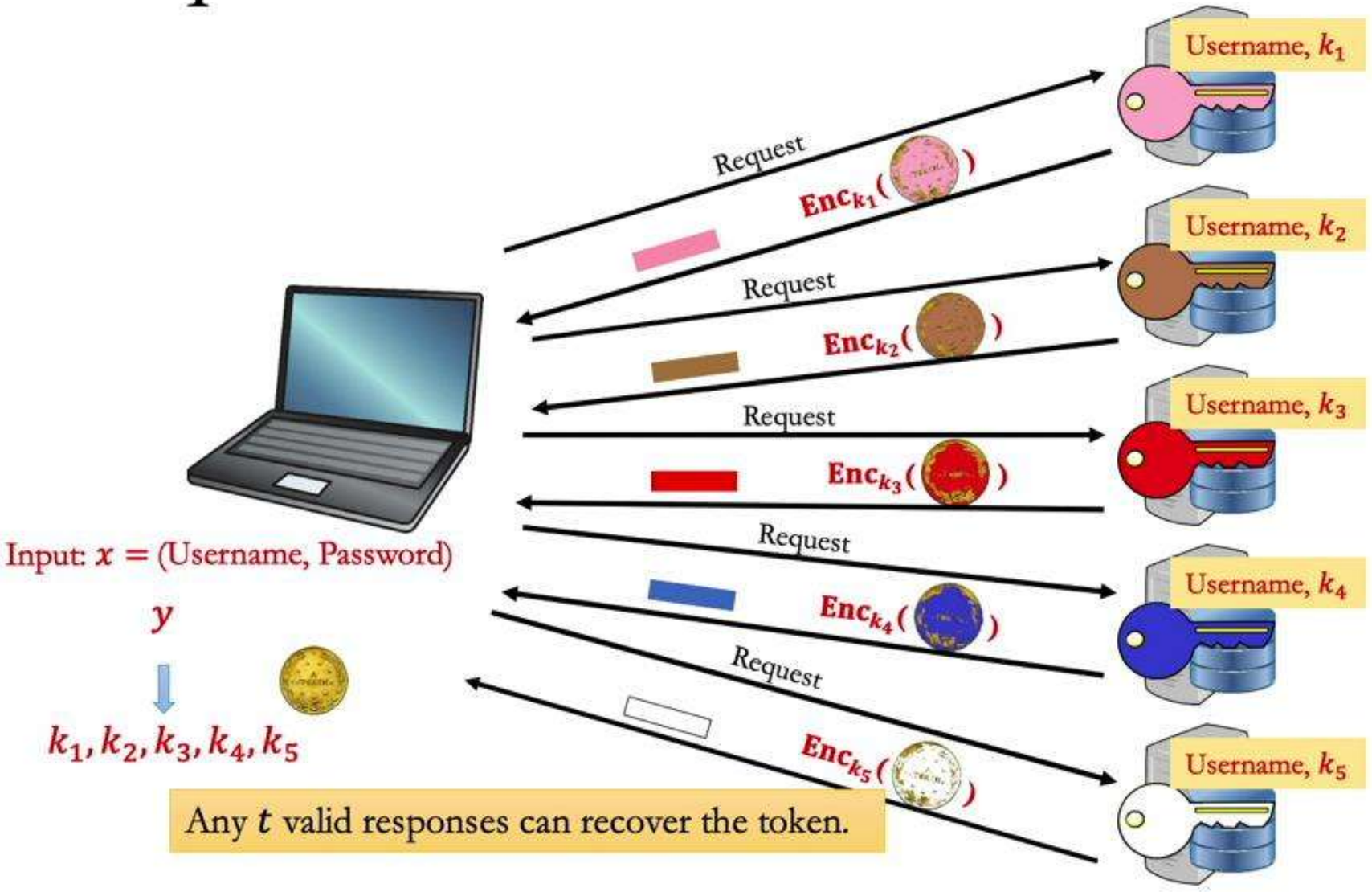
Request for Token – A Fix



Request for Token – A Fix

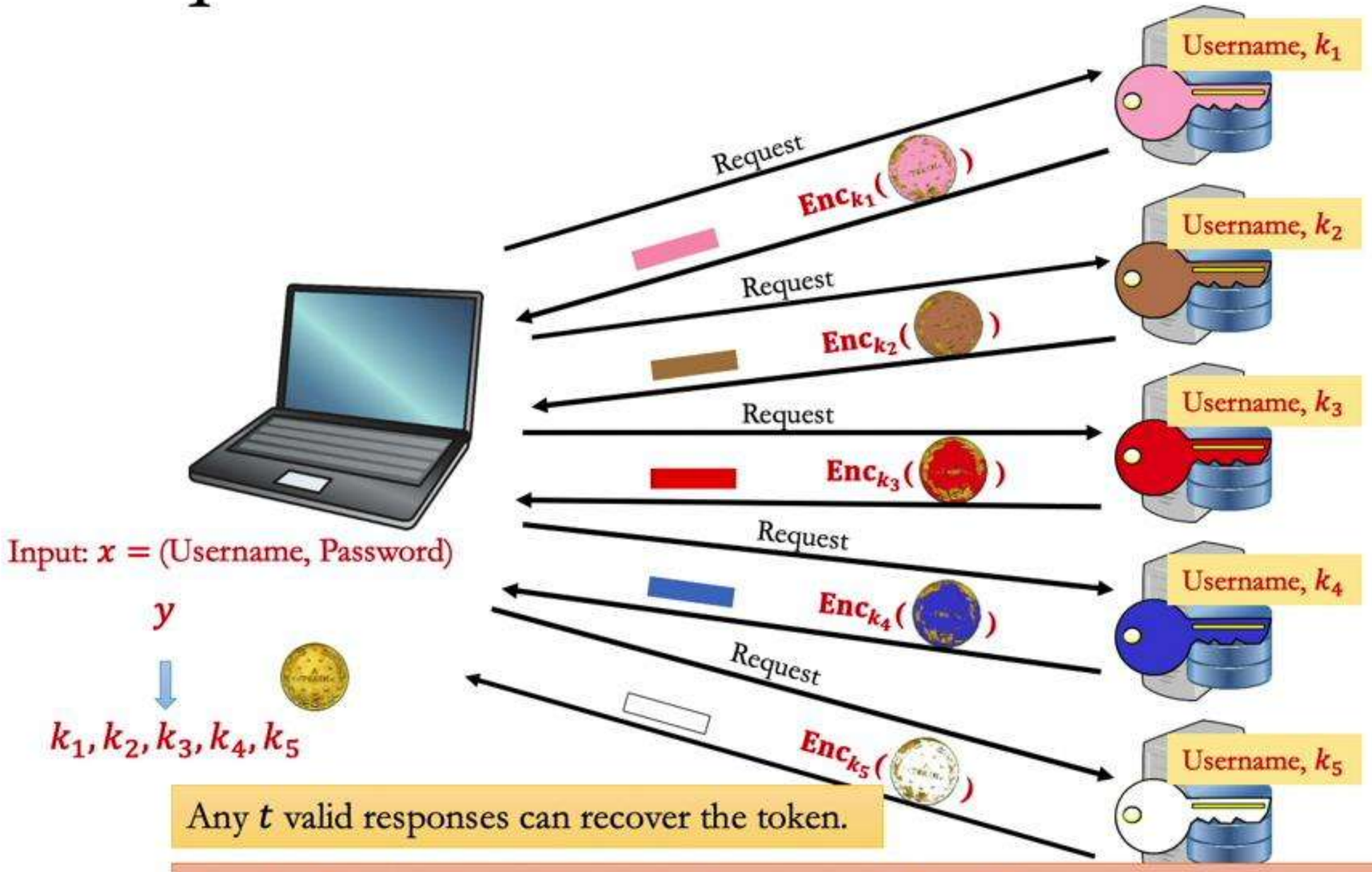


Request for Token – A Fix



Any t valid responses can recover the token.

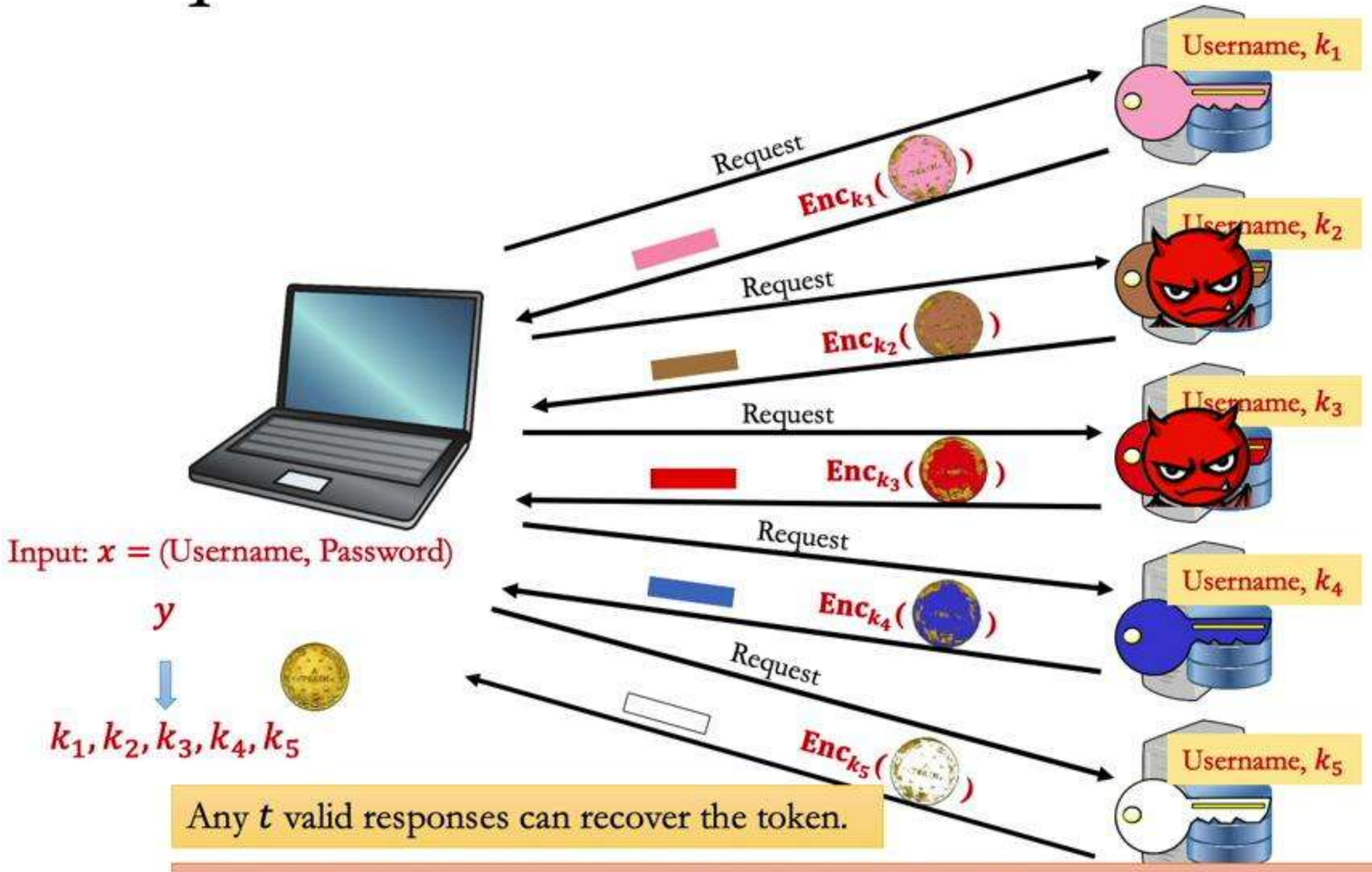
Request for Token – A Fix



Any t valid responses can recover the token.

Any $t-1$ servers cannot recover the password (by offline attack) or forge a token?

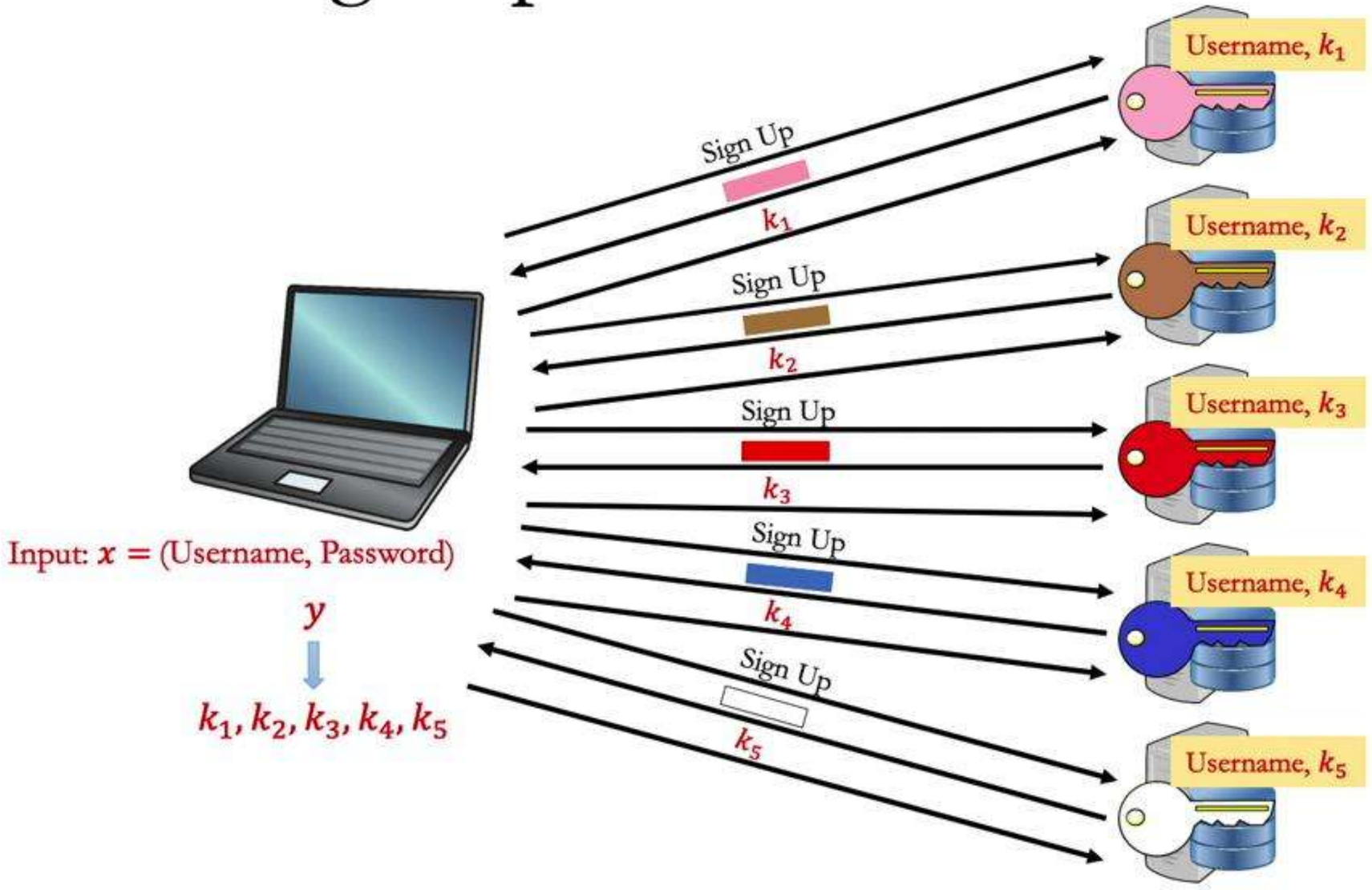
Request for Token – A Fix



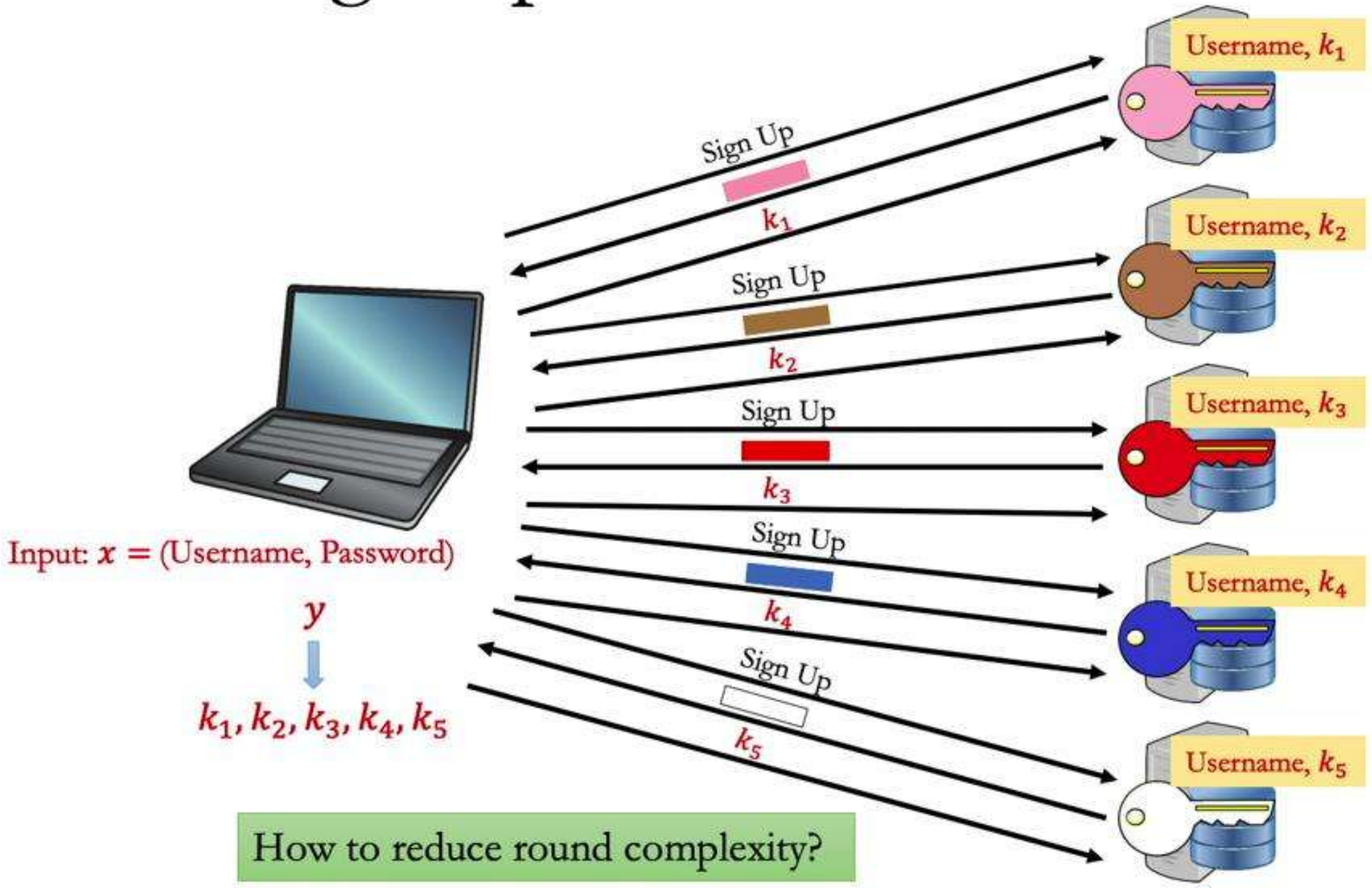
Any t valid responses can recover the token.

Any $t-1$ servers cannot recover the password (by offline attack) or forge a token?

User Sign Up


Username, k_1
Sign Up
k_1
Username, k_2
Sign Up
k_2
Username, k_3
Sign Up
k_3
Username, k_4
Sign Up
k_4
Username, k_5
Sign Up
k_5
Input: x = (Username, Password)
y
k_1, k_2, k_3, k_4, k_5

User Sign Up



How to reduce round complexity?

User Sign Up – Minimal Interaction

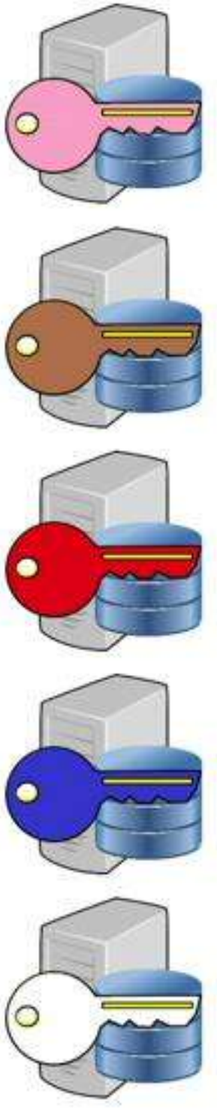
User Sign Up – Minimal Interaction

TOPRF key

User Sign Up – Minimal Interaction

TOPRF key

Input: x = (Username, Password)



User Sign Up – Minimal Interaction

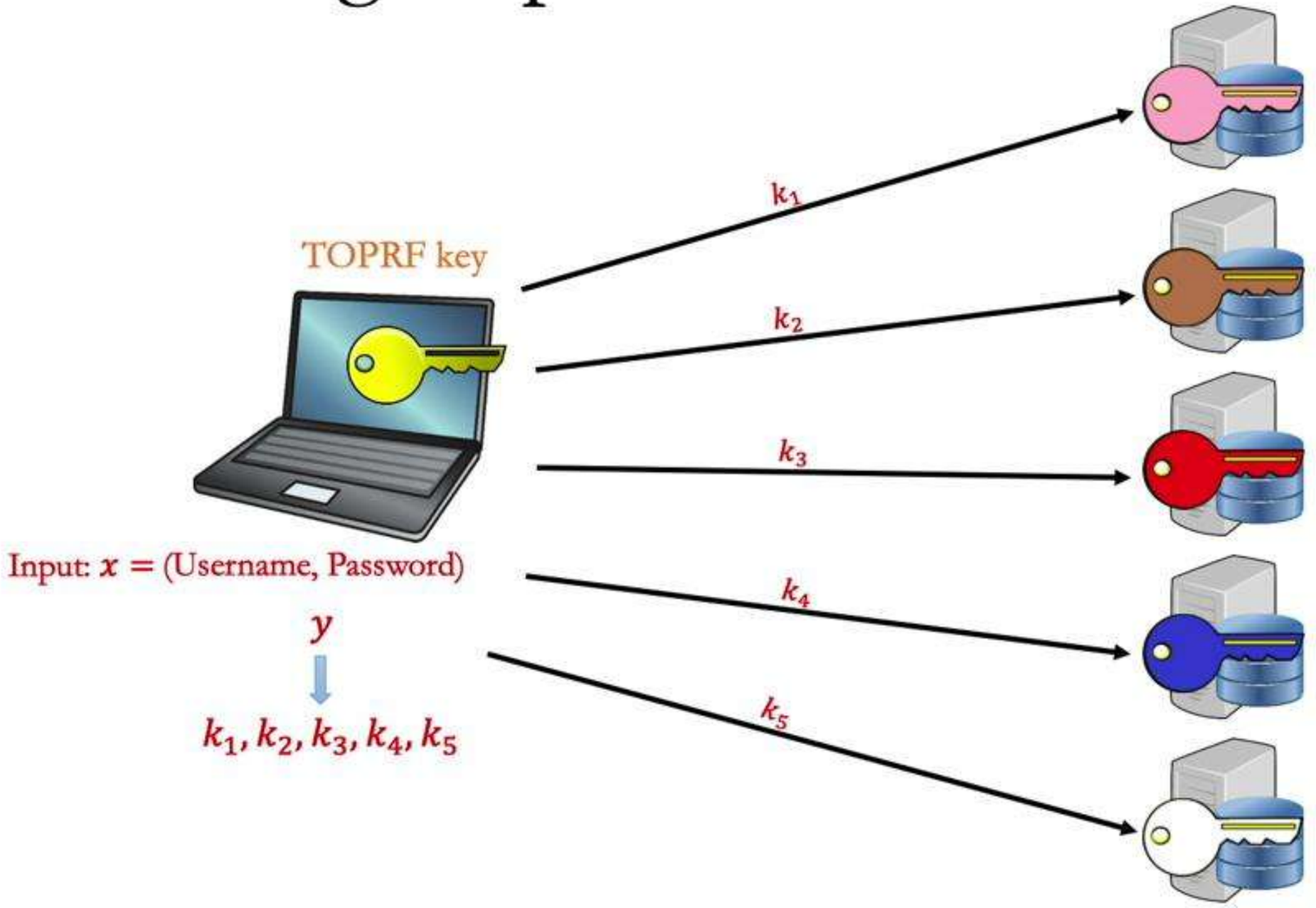
TOPRF key

Input: x = (Username, Password)

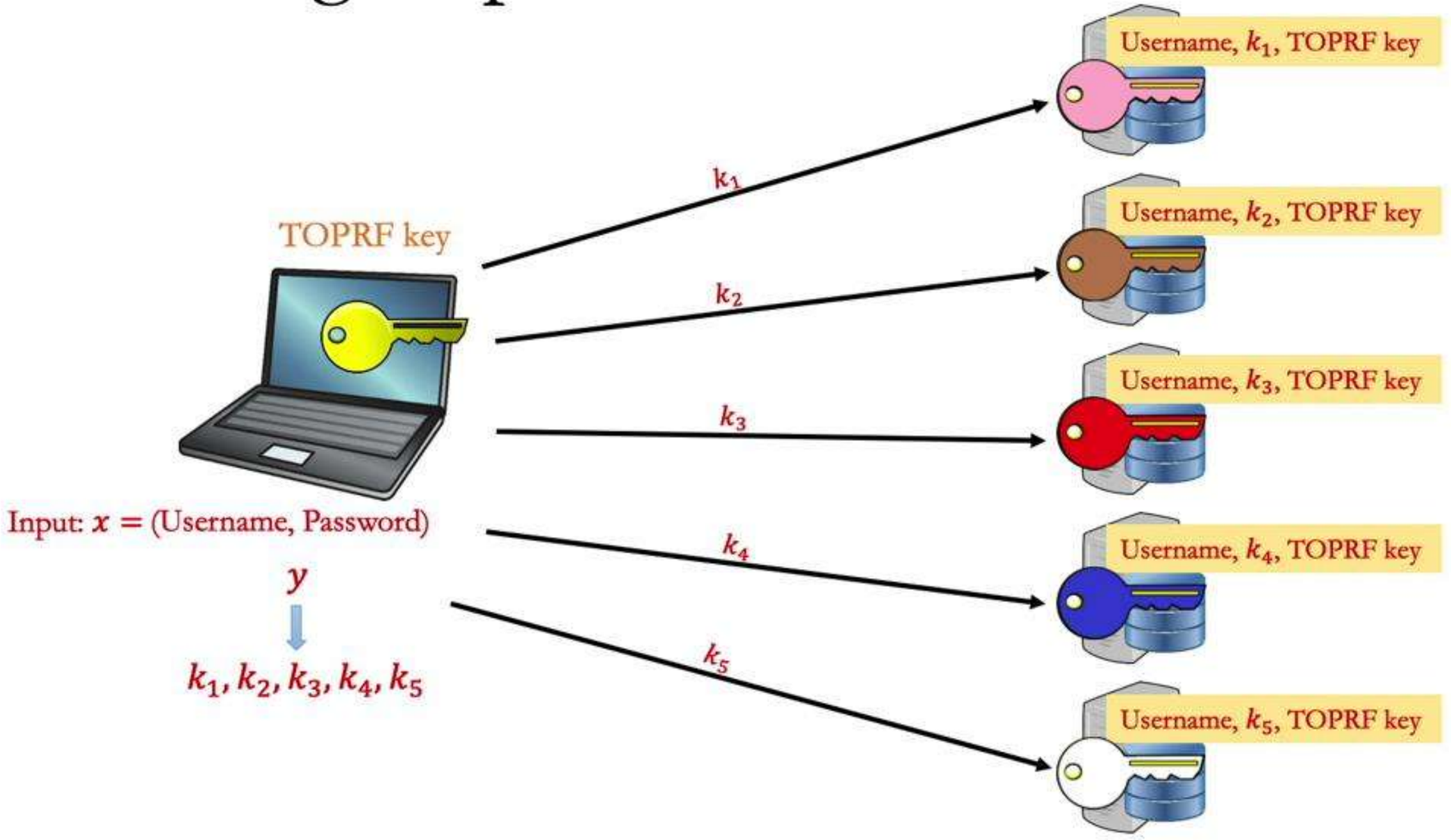
y

k_1, k_2, k_3, k_4, k_5

User Sign Up – Minimal Interaction



User Sign Up – Minimal Interaction



Overview

- What problem is PASTA trying to solve?
- How does PASTA work?
- Is it practical?

Implementation

Implementation

- Non-Interactive Threshold Token Generation (NITTG) for
  - Symmetric key based MAC
  - Public key (DDH) based MAC
  - RSA based digital signature (first implementation)
  - Pairing based digital signature (first implementation)

Implementation

- Non-Interactive Threshold Token Generation (NITTG) for
  - Symmetric key based MAC
  - Public key (DDH) based MAC
  - RSA based digital signature (first implementation)
  - Pairing based digital signature (first implementation)

- Threshold Oblivious Pseudorandom Function (TOPRF)

Implementation

- Non-Interactive Threshold Token Generation (NITTG) for
  - Symmetric key based MAC
  - Public key (DDH) based MAC
  - RSA based digital signature (first implementation)
  - Pairing based digital signature (first implementation)
- Threshold Oblivious Pseudorandom Function (TOPRF)
- PASsword-based Threshold Authentication (PASTA)

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

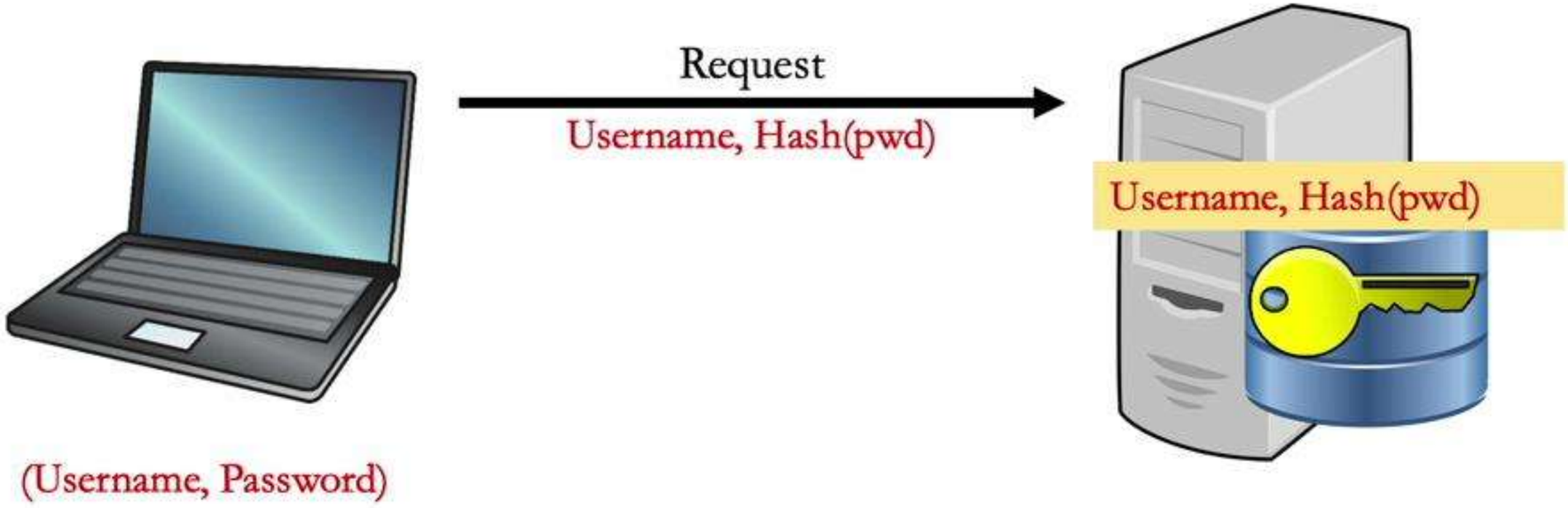
Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

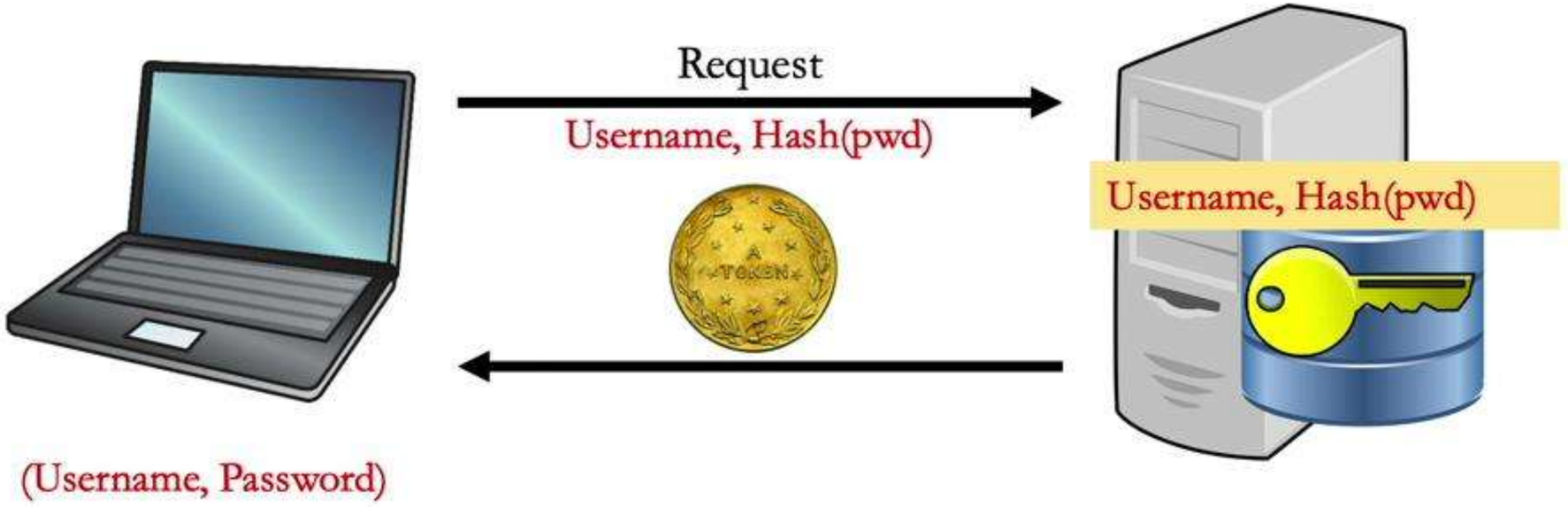
Plain Setting

(Username, Password)

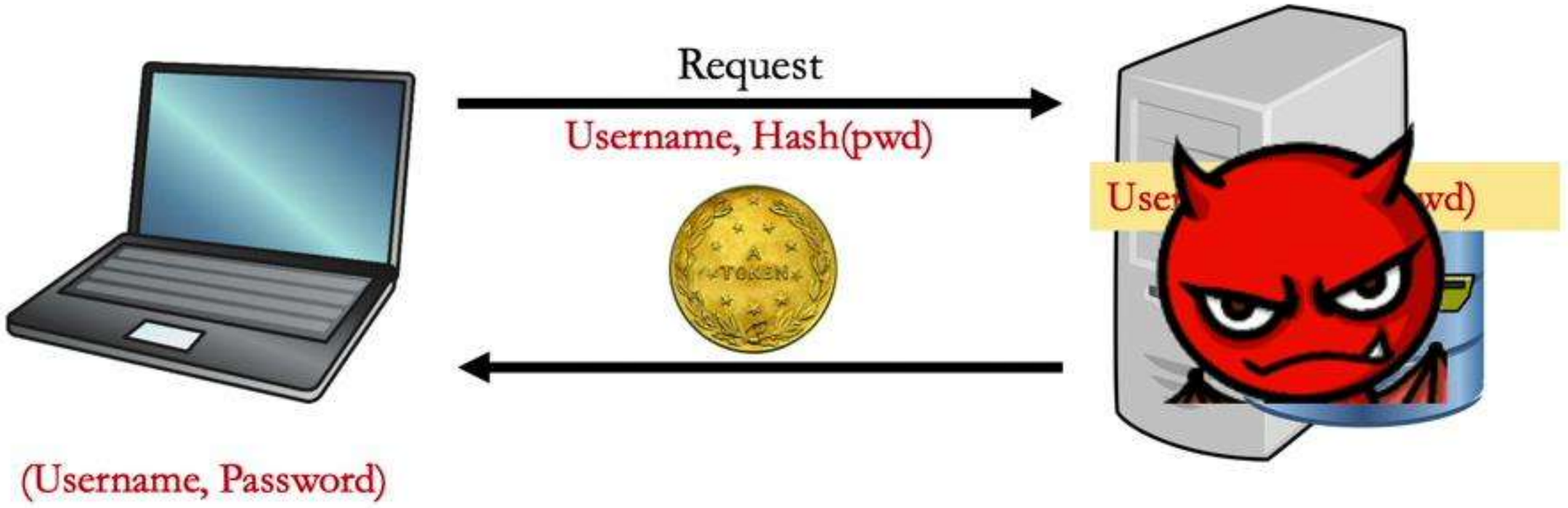
Plain Setting



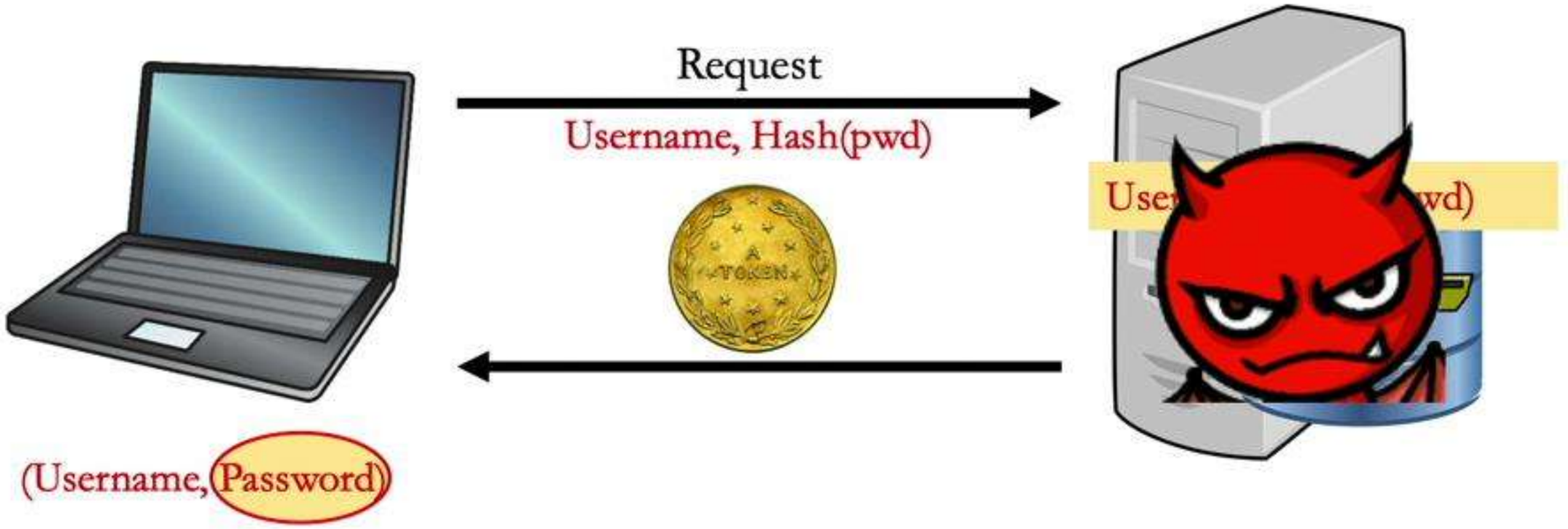
Plain Setting



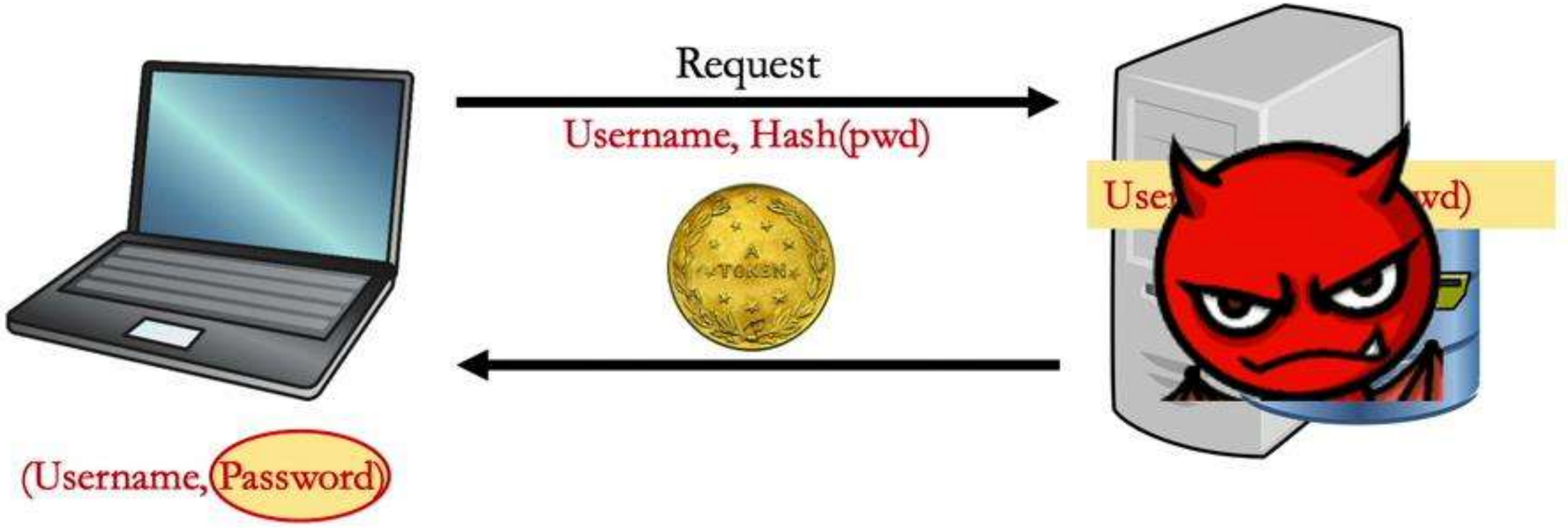
Plain Setting



Plain Setting



Plain Setting



What's the extra cost of protecting secret key and password?

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

Experiment

We showed public key operations are theoretically necessary to achieve our goal.

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1 ⟺	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	80.2	81.4	81.4	81.7	81.4	81.7	82.7	81.4	82.4	84.2
Public-MAC	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
Pairing-Sig	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
RSA-Sig	91.3	94.6	94.6	95.0	94.6	95.3	98.6	94.5	96.9	102.8

Token request performance (ms) for an average of 10,000 token requests

WAN network: 40Mbps, 80ms RTT latency

Experiment

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	80.2	81.4	81.4	81.7	81.4	81.7	82.7	81.4	82.4	84.2
Public-MAC	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
Pairing-Sig	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
RSA-Sig	91.3	94.6	94.6	95.0	94.6	95.3	98.6	94.5	96.9	102.8

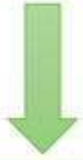
Token request performance (ms) for an average of 10,000 token requests

WAN network: 40Mbps, 80ms RTT latency

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	80.2	81.4	81.4	81.7	81.4	81.7	82.7	81.4	82.4	84.2
Public-MAC	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
Pairing-Sig	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
RSA-Sig	91.3	94.6	94.6	95.0	94.6	95.3	98.6	94.5	96.9	102.8

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	0.1	1.3	1.3	1.6	1.3	1.6	2.7	1.3	2.3	4.1
Public-MAC	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
Pairing-Sig	0.4	1.7	1.7	2.1	1.7	2.1	3.5	1.7	3.0	5.4
RSA-Sig	11.3	14.5	14.5	15.0	14.6	15.1	18.6	14.5	16.8	22.6



+ 80ms

(n, t)	plain	(2,2)	(3,2)	(3,3)	(6,2)	(6,3)	(6,6)	(10,2)	(10,5)	(10,10)
Sym-MAC	80.2	81.4	81.4	81.7	81.4	81.7	82.7	81.4	82.4	84.2
Public-MAC	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
Pairing-Sig	80.4	81.8	81.8	82.2	81.8	82.2	83.6	81.9	83.1	85.4
RSA-Sig	91.3	94.6	94.6	95.0	94.6	95.3	98.6	94.5	96.9	102.8

Our round complexity is optimal!

Time Breakdown

	plain	(10,2)	Server	Client	(10,6)	Server	Client	(10,10)	Server	Client
Sym-MAC	0.1	1.3	0.2	1.0	2.7	0.2	2.3	4.1	0.2	3.7
Public-MAC	0.4	1.7	0.4	1.2	3.5	0.4	2.9	5.4	0.4	4.6
Pairing-Sig	0.4	1.7	0.4	1.2	3.5	0.4	2.9	5.4	0.4	4.6
RSA-Sig	11.3	14.5	11.4	3.0	18.5	11.5	6.8	22.6	11.5	10.7

Token request performance (ms) for an average of 10,000 token requests

LAN network: 10Gbps, 0.1ms RTT latency

Summary

- What problem is PASTA trying to solve?

- How does PASTA work?

- Is it practical?

Summary

- What problem is PASTA trying to solve?
  - Password-based threshold authentication
  - Protect both secret key and passwords
- How does PASTA work?
- Is it practical?

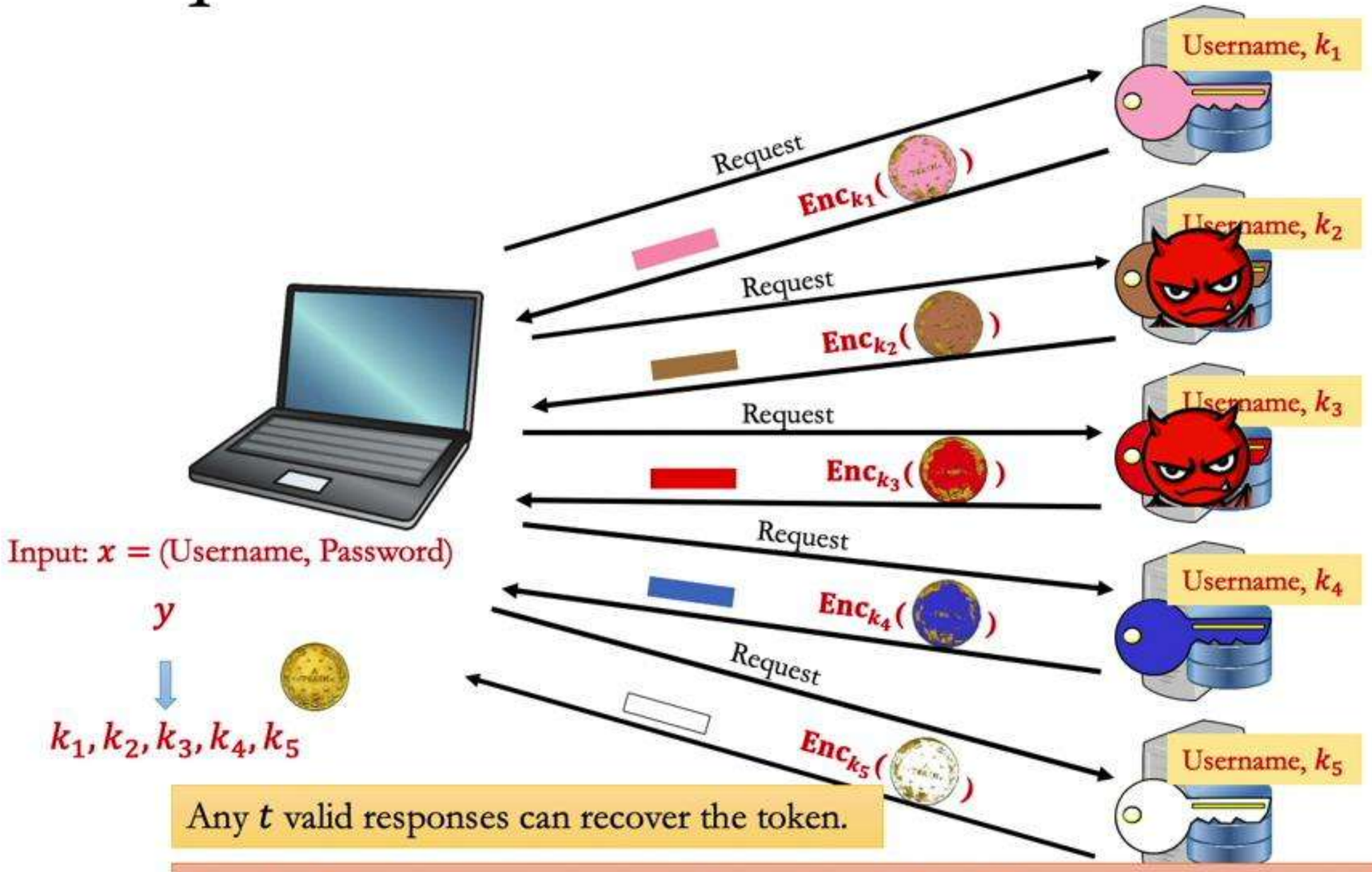
Summary

- What problem is PASTA trying to solve?
  - Password-based threshold authentication
  - Protect both secret key and passwords

- How does PASTA work?
  - Generic construction from NITTG + TOPRF
  - Framework for various types of tokens

- Is it practical?

Request for Token – A Fix



Any t valid responses can recover the token.

Any $t-1$ servers cannot recover the password (by offline attack) or forge a token?