Renewal of the Remote Maintenance System for the SPring-8 Control System

T. SUGIMOTO and T. SAKAMOTO
JASRI/SPring-8, JAPAN

Overview

- Quick Review of Cryptography

- Introduction of SPring-8 and SACLA
- Overview of WARCS (version 1)
  - Problem: Man-in-the-Middle Vulnerability
- Development of New WARCS (version 2)
- Summary

Man-in-the-Middle Attack

Authentication

Quick Review of Cryptography

Character

Alice
Sender (or Client)

Bob
Receiver (or Server)

Eve
Eavesdropper

Mallory
Malicious Attacker

Man-in-the-Middle Attack

Send a clear-text letter from Alice to Bob

Alice

Bob

Man-in-the-Middle Attack

Clear-text letter can be read by Eve

Alice

!

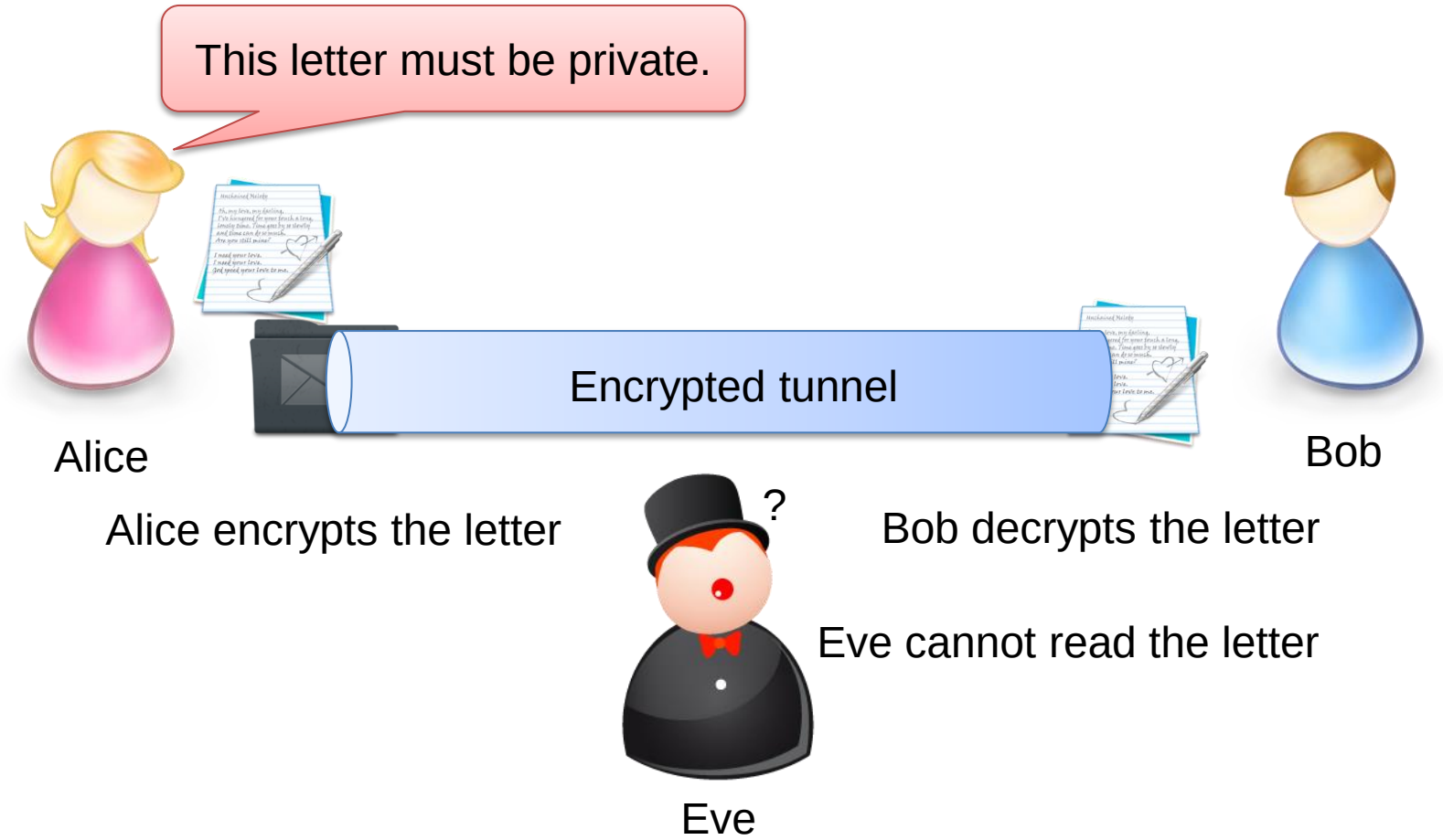
Eve can read the letter

Eve

Bob

Man-in-the-Middle Attack

Encrypted letter cannot be read by Eve



When encryption and decryption is point-to-point and performed like “flow”, we call the flow as “Encrypted tunnel”.

Man-in-the-Middle Attack

Encrypted letter requires algorithm and key

Alice and Bob must be agree on the encryption algorithm and key.

Alice

Bob

Alice encrypts the letter

Bob decrypts the letter

Man-in-the-Middle Attack

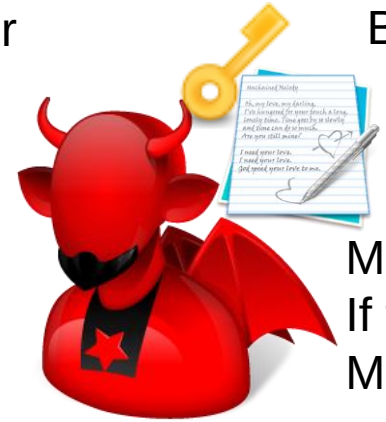
If Mallory copy the key, the letter can be read by Mallory

Alice

Bob

Alice encrypts the letter

Bob decrypts the letter



Mallory

Mallory may capture the key.
If the key is copied by Mallory,
Mallory can alternate the letter.

Man-in-the-Middle Attack

How to share the key?

It is important to share the encryption key safely.

Most popular technique is Diffie-Hellman key exchange.
Details are skipped in this talk.

Server and Client authentication

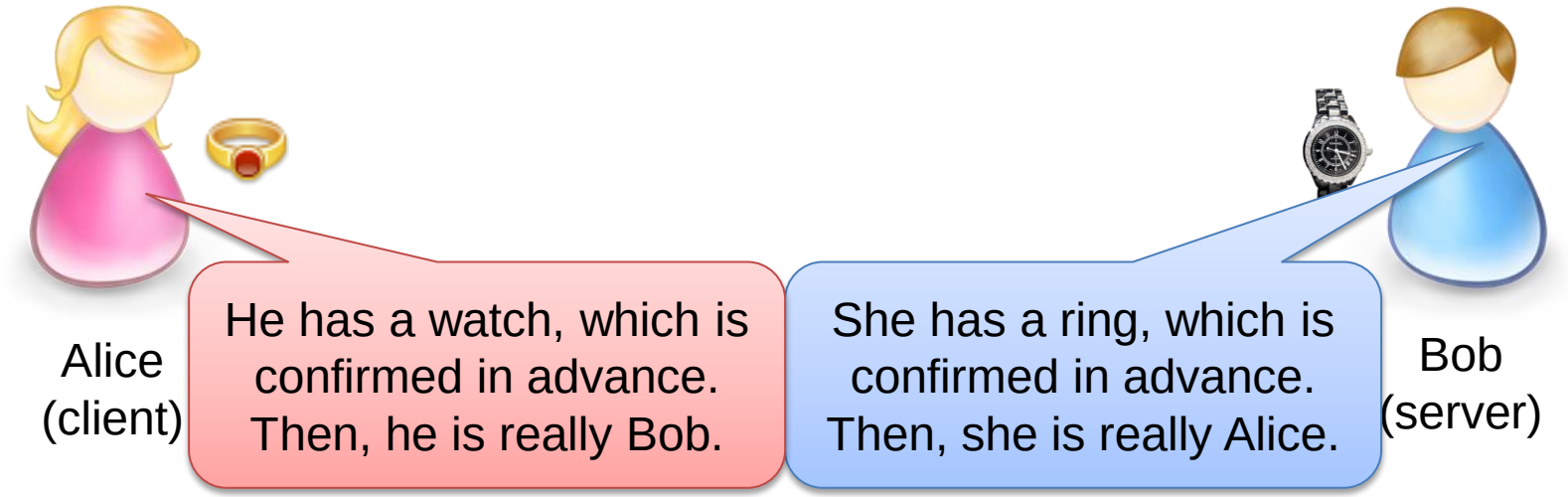
Alice
(client)

Bob
(server)

Is she really Alice?

Is he really Bob?

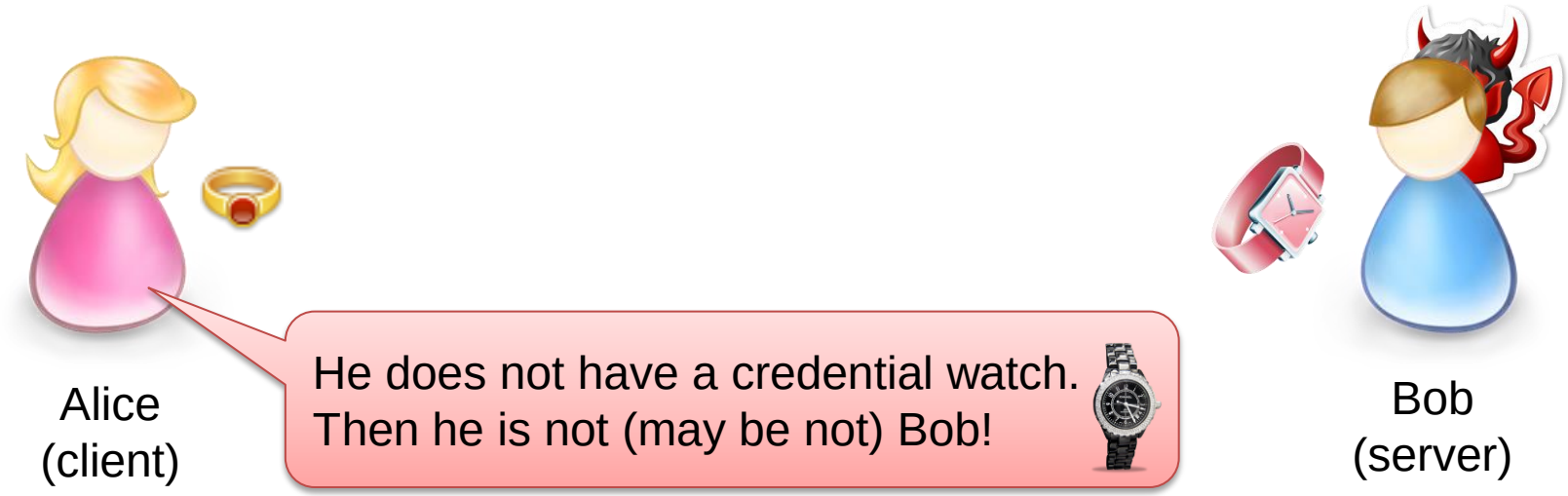
Server and Client authentication



Is she really Alice?

Is he really Bob?

Server and Client authentication



Is she really Alice?

She has a credential ring.
Then she is really Alice.

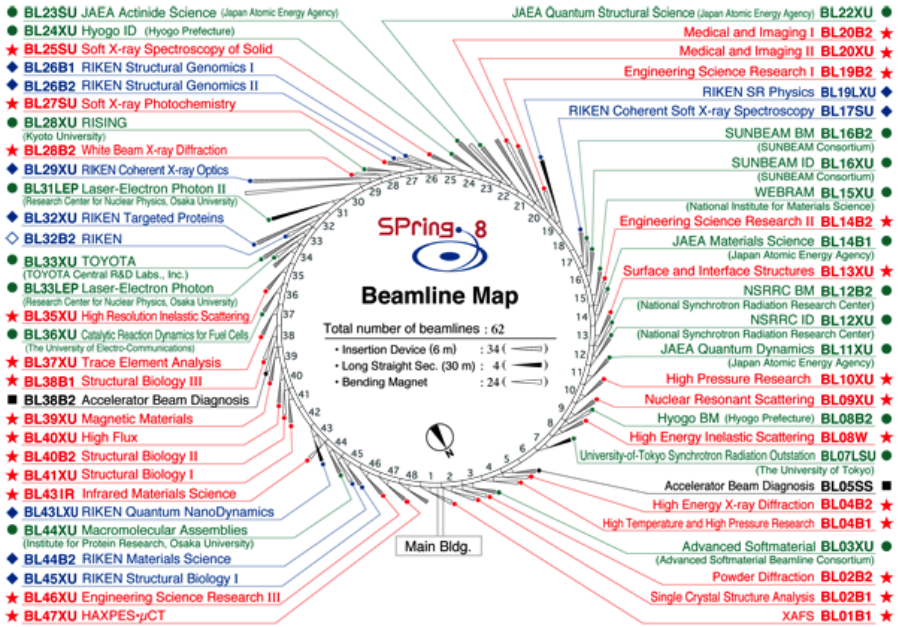
Is he really Bob?

Credentials may not be object.
The credentials may be signature, password, public-key, etc..

Introduction of SPring-8 and SACLA

SPring-8
Synchrotron Radiation Facility

SPring-8 is a synchrotron radiation facility, located at west region of JAPAN.

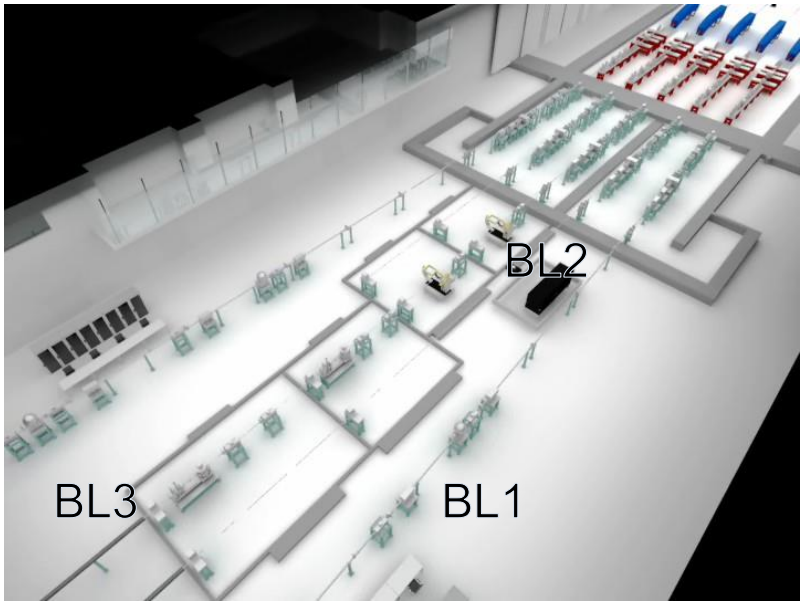


About 60 beamlines are in operation.

Operation time is >5,000 hour/year.
Gross number of experimental user is >10,000 people/year.

SACLA
X-ray Free Electron Laser Facility

SACLA is a X-ray free electron laser facility, located in the SPring-8 site.

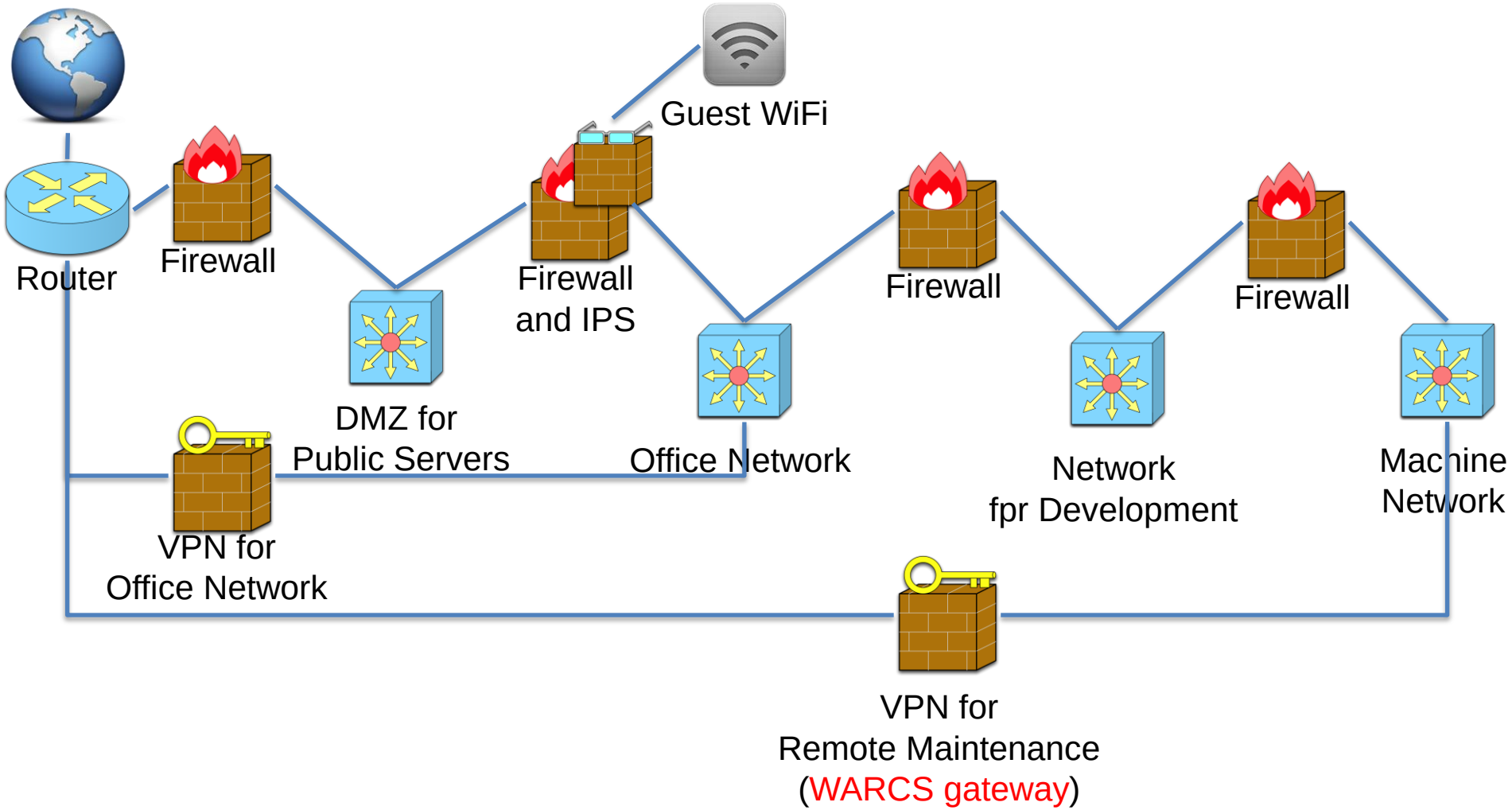


2 beamlines (BL1 and BL3) have been constructed, and BL3 is in operation. Additional beamline (BL2) is under construction.

Operation time is 7,000 hour/year.

Schematic View of SPring-8 and SACLA Network System

Multi-layered network depends on each security level.



Requirement of Remote Access to Machine Control System

- Many experiments are proposed and approved. Therefore, the operation time of SPring-8 and SACLA is so long.

- Stable operation is important to satisfy users' experimental requirement.
  - When machine trouble occurs, rapid recovery is required, even if expert is absent from SPring-8 site.

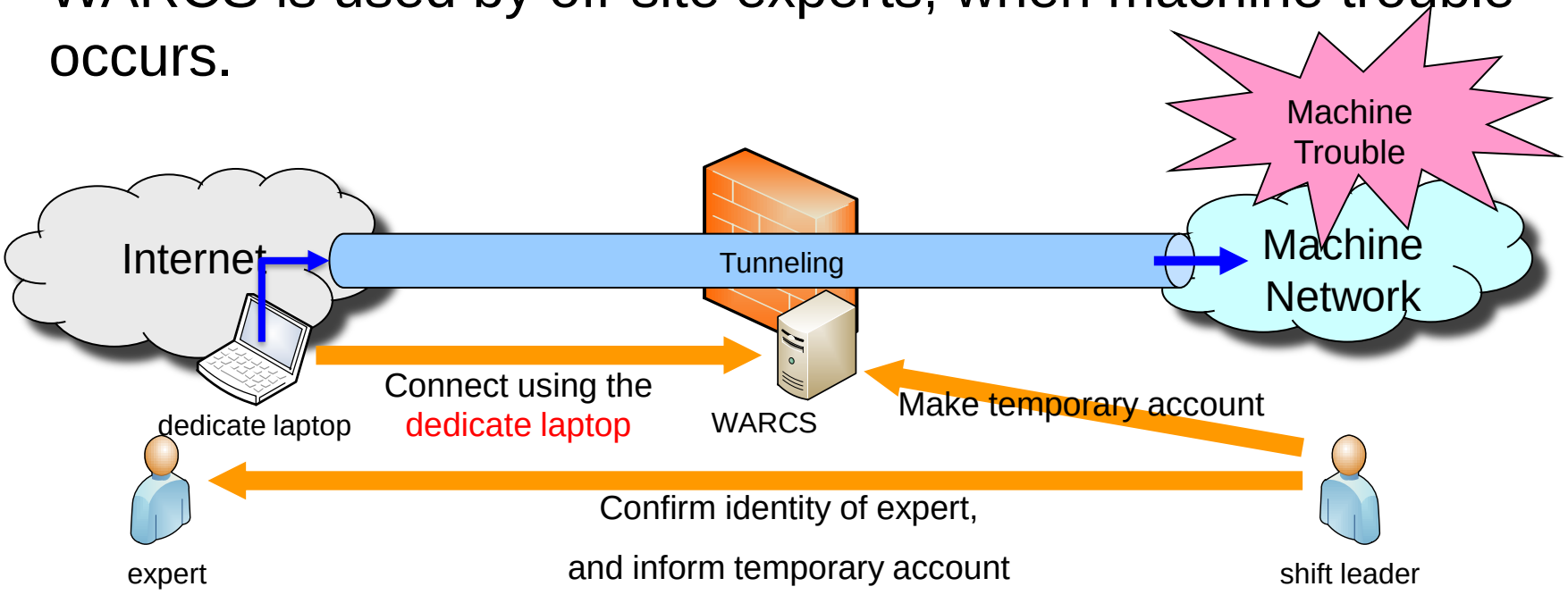
- To realize remote access to the machine control system, we developed dedicate remote access system, named WARCS (Wide Area Remote Control System).[1]

[1] A. Yamashita and Y. Furukawa, Proceedings of ICALEPCS 2005.

Overview of WARCS (version 1)

What is WARCS?

- WARCS is a remote access system for SPring-8.
- WARCS is used by off-site experts, when machine trouble occurs.



- WARCS version 1 (WARCSv1) is assembled using open-sourced database (SQLite) and tunneling application (Zebedee).

Zebedee

- Zebedee is a IP tunneling software
  - http://www.winton.org.uk/zebedee/
  - Zlib compression
  - Encryption method is based on Blowfish algorithm.
  - Diffie-Hellman key agreement (exchange).

- Please remember, Zebedee does not have authentication method. Zebedee have key-exchange capability.
  - “Authentication” is not equal to “key exchange”.

Requirements of the WARCS

- WARCS is not a mere VPN. WARCS must be satisfy operational and legal requirements.
  - Encrypted tunneling
    - Virtual private network between dedicate laptop and machine network
  - Personal identification and authentication
    - Personal identification (call phone, then confirm identity of expert)
    - Login authentication (one-time password, informed from shift leader)
  - Permissions by SHIFT LEADER
    - Shift leader gives a control permission to expert
    - Shift leader can disconnect the tunnel any time

Implementation of WARCSv1

	WARCSv1
VPN Scheme	Zebedee
Remote Terminal	Dedicate Laptop (Zebedee is installed in advance)
Authentication	one-time password
Encryption	depends on Zebedee (Blowfish)
Authorization	by Shift Leader (confirm identity via phone)

Problem of WARCSv1

- Recently, we faced on problem using WARCSv1
  - Zebedee is not permitted
    - Zebedee is regarded as a technology to be exploited, like a Tor.
    - Zebedee is not permitted to use in many situations; hotels, universities nor institutes (including SPring-8).
  - Dedicate laptop is required
    - Expert want to use his custom laptop.
      - Not only Windows, but also Mac.
      - The dedicate laptop is Windows only, and installed software are limited.
    - If expert forget to bring the dedicate laptop, he cannot use the WARCS.

and

- Vulnerability is found in the WARCSv1.

WARCSv1 is Insecure!

- Authentication process of WARCSv1
  - Client PC sends one-time password (OTP) via non-encrypted HTTP.
  - Client PC also sends public key for authentication. However, the corresponding public/private key pair are generated in the client PC itself.

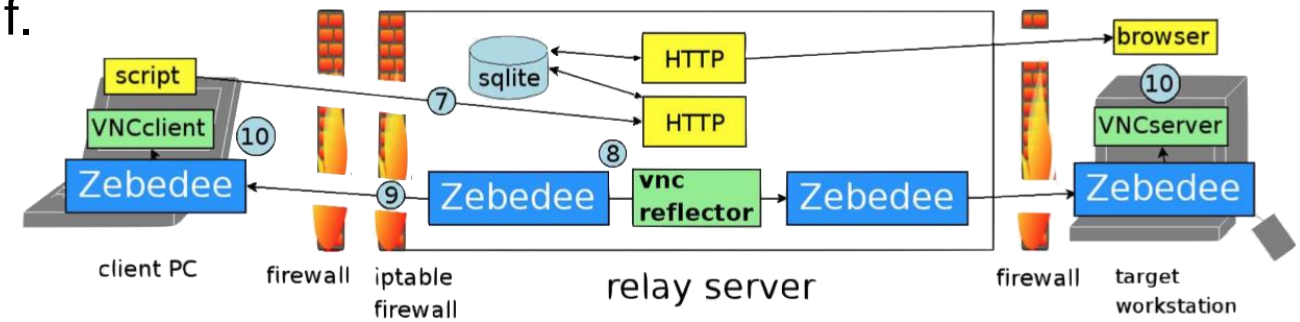


Fig4. Establish connection.

The expert connects a notebook PC to the internet and starts a client script written in python (Fig3-6). The script accepts an one time password from the experts and generates a secret and a public key for zebedee. The script sends the one time password and the public key to the web sever by http protocol (Fig4-7). If the password matches to the one store in the database the cgi-bin process start up zebedee process with public key authentication (Fig4-8). Those http communications are not encrypted. The cgi-bin process also adds ip-address and zebedee port to the iptable chain to allow communication through relay server's firewall (Fig4-9). In short, it digs a tunnel through the firewall. The zebedee server for outside client only accepts communication from the computer which has private key which is matched to the public key.

WARCSv1 is Insecure!

- Authentication process of WARCSv1
  - Client PC sends one-time password (OTP) via non-encrypted HTTP.
  - Client PC also sends public key for authentication. However, the corresponding public/private key pair are generated in the client PC itself.
- What is wrong?
  - Public-key authentication is not running
    - Public/private key pair are generated in the client PC instantly. There is no secure method to store private key with the server.
  - Wrong use of OTP
    - OTP authentication is vulnerable against Man-in-the-Middle (MITM) attack.
    - The OTP authentication must be used with encrypted tunnel. However, no encrypted tunnel is established at the authentication process of WARCSv1.

Proof of Vulnerability in the WARCSv1

Temporary account

外部ログイン制御

- 外部ログイン パスワードの発行
- 外部ログイン状況

Akihiro Yamashita <aki@spring8.or.jp>
Last modified: Fri Sep 12 09:52:41 2003

select count(*) from login where loginname="sakamoto"
select max(cport),min(cport) from login
[(8083.0, 8081.0)] insert into login values ('20V.e8mbjLSDY',0,1375666516,'sakamoto','',8080,'10.10.62.88',23)

OK!

user	sakamoto
host	vm-test08
port	8080
host port	23
password	2037839

login name	use_flg	authorozed time	client	host	Action	log
test01	Now logon	2013/08/05 09:29:08	10.10.252.253:8081	vm-test08.spring8.or.jp:23	delete	log
test02	Now logon	2013/08/05 09:31:11	10.10.252.253:8082	vm-test08.spring8.or.jp:23	delete	log
test03	Now logon	2013/08/05 09:44:06	accsunray3.spring8.or.jp:8083	vm-test08.spring8.or.jp:23	delete	log

host port 23
password 2037839

Proof of Vulnerability in the WARCSv1

Packet capture (on client PC)

Intel(R) 82567LM Gigabit Network Connection: Capturing - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: ip.addr == 10.10.58.104 Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
8735	378.152837	10.10.58.104	10.10.63.255	UDP	Source port: 49164 Destination port: hlserver
9294	405.172268	10.10.58.104	239.255.255.250	IGMP	V2 Membership Report / Join group 239.255.255.250
9490	409.805508	10.10.58.104	224.0.0.252	IGMP	V2 Membership Report / Join group 224.0.0.252
9588	414.832606	10.10.58.104	255.255.255.255	UDP	Source port: 49164 Destination port: hlserver
9663	418.854267	10.10.58.104	10.10.63.255	UDP	Source port: 49164 Destination port: hlserver
10351	453.150071	10.10.58.104	10.10.27.1	DNS	Standard query A try-warcs.spring8.or.jp
10352	453.150507	10.10.27.1	10.10.58.104	DNS	Standard query response A 10.10.59.91
10357	453.172481	10.10.58.104	10.10.59.91	TCP	50819 > irdmi [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
10358	453.172934	10.10.59.91	10.10.58.104	TCP	irdmi > 50819 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=6
10360	453.173492	10.10.58.104	10.10.59.91	TCP	50819 > irdmi [ACK] Seq=1 Ack=1 Win=65700 Len=0
10361	453.174105	10.10.58.104	10.10.59.91	TCP	50819 > irdmi [PSH, ACK] Seq=1 Ack=1 Win=65700 Len=130
10362	453.174331	10.10.59.91	10.10.58.104	TCP	irdmi > 50819 [ACK] Seq=1 Ack=131 Win=6912 Len=0
10373	453.622035	10.10.59.91	10.10.58.104	TCP	irdmi > 50819 [PSH, ACK] Seq=1 Ack=131 Win=6912 Len=1134
10374	453.626753	10.10.59.91	10.10.58.104	TCP	irdmi > 50819 [PSH, ACK] Seq=1135 Ack=131 Win=6912 Len=262
10375	453.626796	10.10.58.104	10.10.59.91	TCP	50819 > irdmi [ACK] Seq=131 Ack=1397 Win=64304 Len=0
10376	453.629106	10.10.59.91	10.10.58.104	TCP	irdmi > 50819 [FIN, ACK] Seq=1397 Ack=131 Win=6912 Len=0
10377	453.629218	10.10.58.104	10.10.59.91	TCP	50819 > irdmi [ACK] Seq=131 Ack=1398 Win=64304 Len=0
10378	453.629359	10.10.58.104	10.10.59.91	TCP	50819 > irdmi [FIN, ACK] Seq=131 Ack=1398 Win=64304 Len=0

```
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 170
    Identification: 0x03ef (1007)
  ⊞ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
  ⊞ Header checksum: 0x6c88 [correct]
    Source: 10.10.58.104 (10.10.58.104)
    Destination: 10.10.59.91 (10.10.59.91)
⊟ Transmission Control Protocol, Src Port: 50819 (50819), Dst Port: irdmi (8000), Seq: 1, Ack: 1, Len: 130
    Source port: 50819 (50819)
```

```
0000  de ad be ef 29 07 00 23  18 b8 87 44 08 00 45 00   ....)..# ...D..E.
0010  00 aa 03 ef 40 00 80 06  6c 88 0a 0a 3a 68 0a 0a   ....@... l...:h..
0020  3b 5b c6 83 1f 40 fc d4  fc f1 66 8c 01 14 50 18   ;[...@.. ..f...P.
0030  40 29 b2 85 00 00 47 45  54 20 2f 63 67 69 2d 62   @)....GE T /cgi-b
0040  69 6e 2f 61 75 74 68 30  32 2e 70 79 3f 70 61 73   in/auth0 2.py?pas
0050  73 77 6f 72 64 3d 32 30  33 37 38 33 39 26 6b 65   sword=20 37839&ke
0060  79 3d 34 66 66 64 35 31  39 38 62 34 66 62 31 62   y=4ffd51 98b4fb1b
0070  30 62 36 30 65 61 30 66  35 37 62 65 61 62 63 34   0b60ea0f 57beabc4
0080  62 37 38 36 66 63 37 66  32 34 25 32 30 57 41 52   b786fc7f 24%20WAR
0090  43 53 54 45 52 4d 31 31  20 48 54 54 50 2f 31 2e   CSTERM11  HTTP/1.
00a0  30 0d 0a 41 63 63 65 70  74 3a 20 74 65 73 74 2f   0..Accep t: test/
00b0  68 74 6d 6c 0d 0a 0d 0a                            html....
```

Client PC sends clear-text password!

```
62   @)....GE T /cgi-b
73   in/auth0 2.py?pas
65   sword=20 37839&ke
62   y=4ffd51 98b4fb1b
34   0b60ea0f 57beabc4
52   b786fc7f 24%20WAR
2e   CSTERM11  HTTP/1.
2f   0..Accep t: test/
     html....
```

Intel(R) 82567LM Gigabit Network Conn··· | Packets: 17548 Displayed: 200 Marked: 0 | Profile: Default

Review of wrong implementation in WARCSv1

- Misuse of OTP authentication
  - Clear-text OTP is vulnerable to MITM attack.
  - To avoid MITM attack, encrypted tunnel with server identification must be established before sending OTP.
- Misunderstanding of “Secure channel”
  - “Secure channel” mean “encrypted tunnel” AND “server authentication” AND “client authentication”.
  - WARCSv1 can not make secure channel, only encrypted tunnel.
    - Zebedee is not used for authentication, but only for encrypted tunnel by DH key exchanging. “Authentication” is not equal to “key-exchange”.
  - “Encrypted tunnel” established by DH key exchange is vulnerable to MITM attack.
- Public-key authentication of WARCSv1 is non sense.

Therefore, we decided to develop new secure WARCS (version 2).

Development of New WARCS (version 2)

Careful attention to authentication procedure in WARCSv2

- How to establish secure channel
  - What type of encrypted tunnel

    → SSL tunneling
  - How to authenticate server credentials

    → SSL server certificate and public-key authentication
  - How to authenticate client credentials

    → Password authentication issued from shift leader
  - What encryption tunnel is permitted at the many situation.

    → SSL-VPN use 443/tcp (HTTPS). We can use HTTPS in many situations.

SSL-VPN is suitable for WARCSv2.

Implementation of WARCSv2

- We separate components into each functions.
  - Temporary account registration GUI
  - Authentication
  - Tunneling

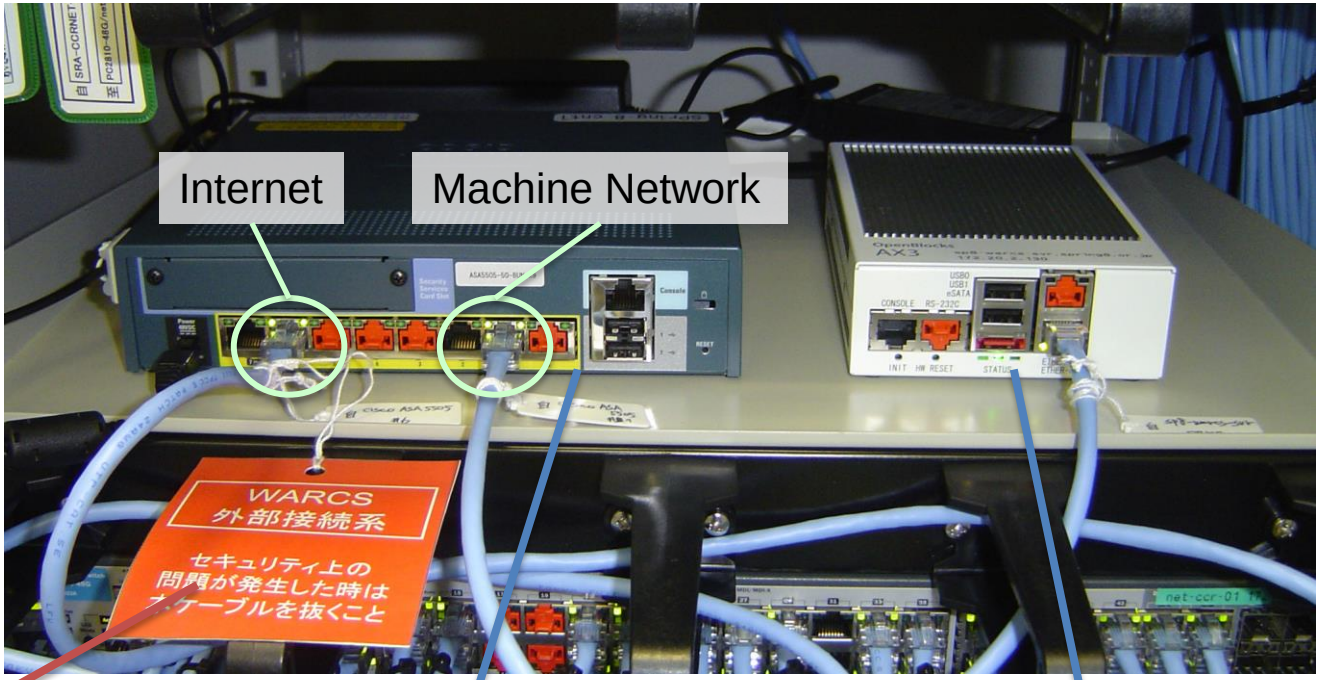
- We use standard AAA framework (RADIUS) for authentication.
  - By using RADIUS as authentication framework, many VPN gateways are compatible to the WARCSv2.

- We choose Cisco ASA5505 as the VPN gateway.
  - Experts can use their laptops including Windows, Mac(, and Linux).

Compare WARCSv1 with WARCSv2

	WARCSv1	WARCv2
VPN Scheme	Zebedee	Standard AAA and VPN
Remote Terminal	Dedicate Laptop (Zebedee is installed in advance)	Expert's Laptop (VPN Client is automatically installed)
Authentication	one-time password	RADIUS PAP (username & password)
Encryption	depends on Zebedee (Blowfish)	SSL-VPN
Authorization	by Shift Leader (confirm identity via phone)	by Shift Leader (confirm identity via phone)

Hardware Components of WARCSv2



Emergency Tag

Cisco ASA5505

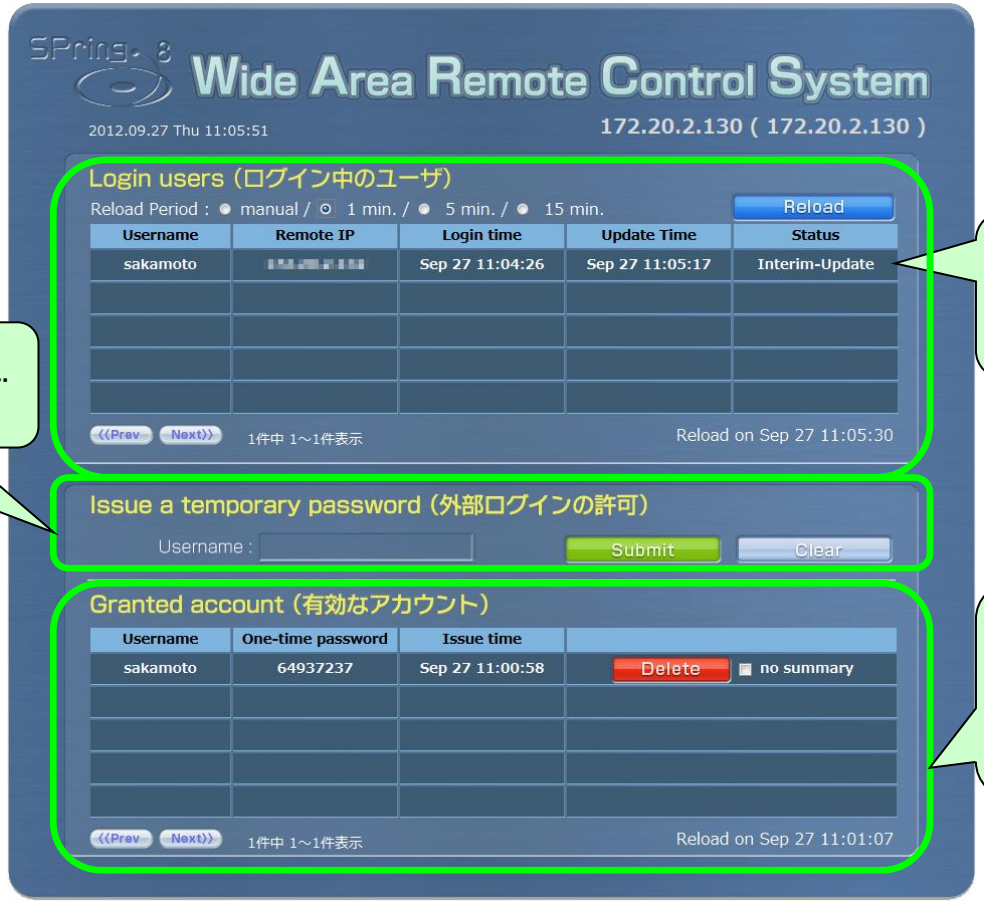
- SSL-VPN

OpenBlocks AX3
(ARM Corex-A9 micro PC)

- RADIUS Server
- Temporary account registration GUI

Account Registration GUI

Since shift leader is not IT expert, we wrapped the RADIUS functions in the single GUI.



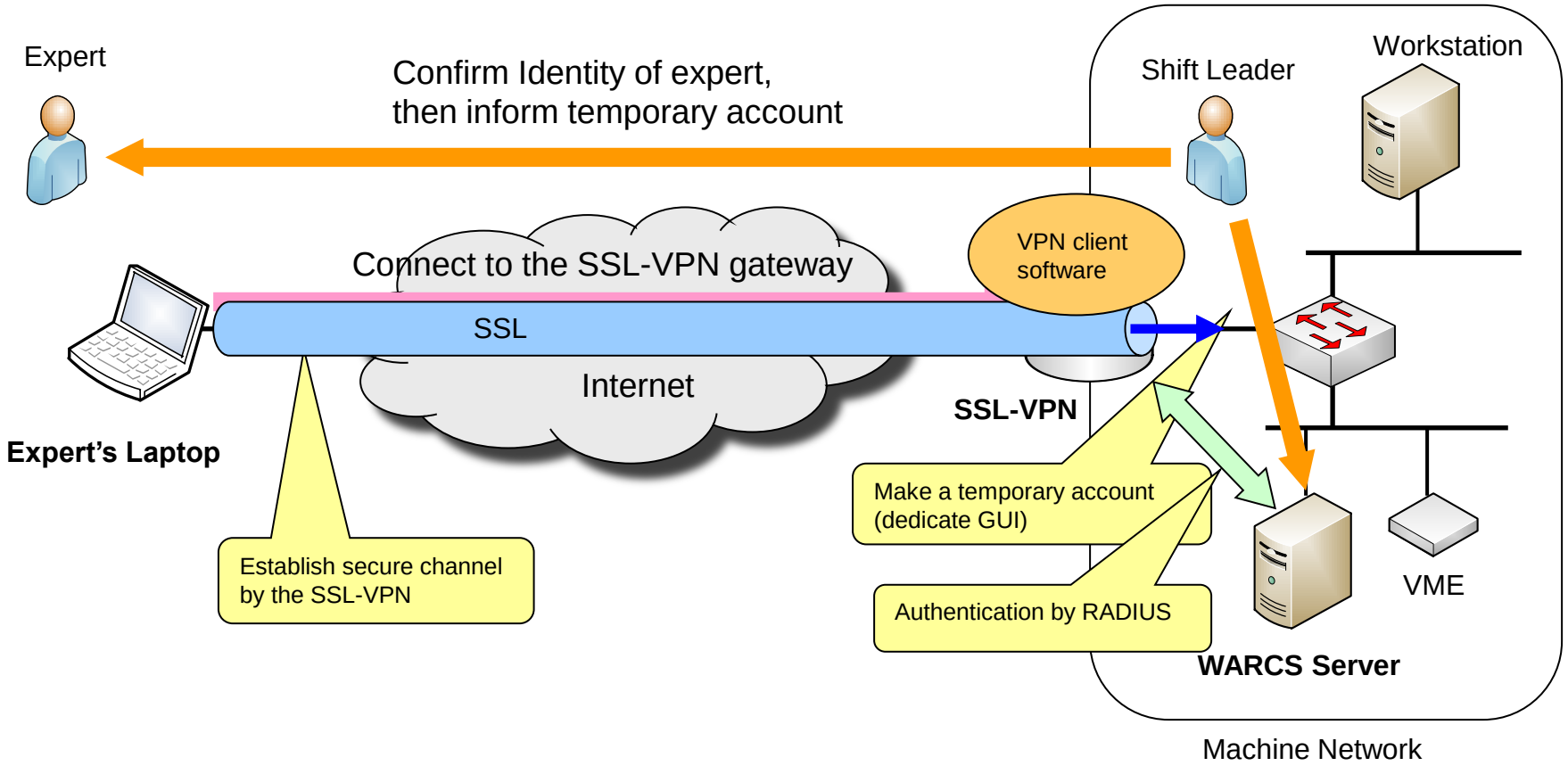
Registration of new account.
(modify /etc/raddb/users)

Active users, connected to the SSL-VPN gateway.
(view RADIUS accounting logs)

List of active accounts.
(view /etc/raddb/users)

and delete account
(modify /etc/raddb/users)

Operation Process of WARCSv2



Connection between the laptop and VPN gateway is server-authenticated encrypted tunnel. Therefore, secure channel is to be established by client authentication. This process is very different from that of WARCSv1.

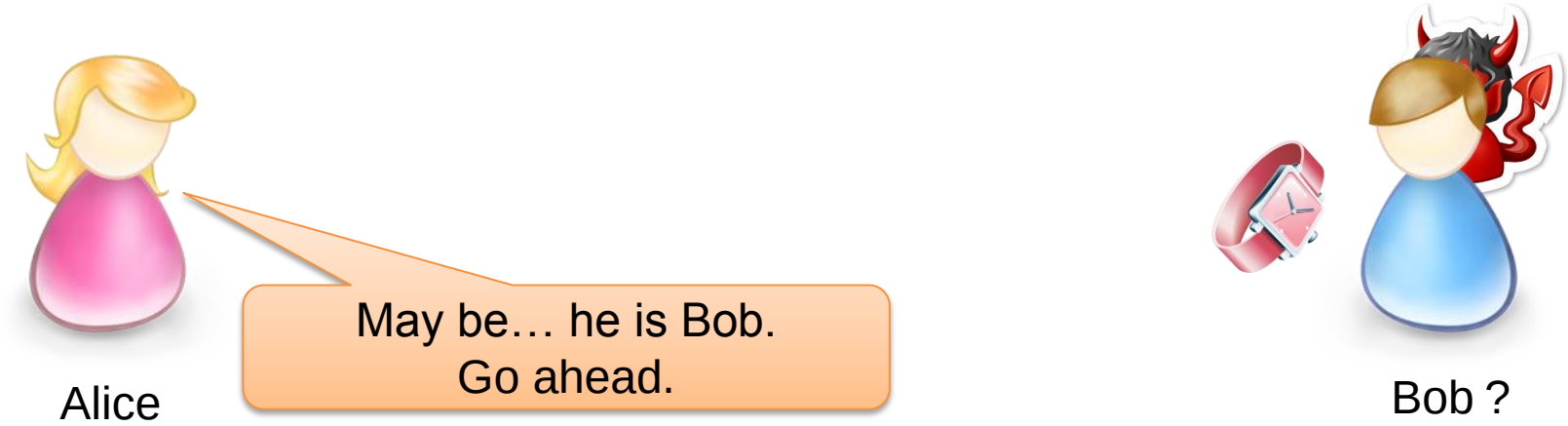
WARCSv2 is in Operation

- WARCSv2 is installed to the SPring-8 control system in October 2012.
- WARCSv2 is also installed to the SACLA control system in January 2013.

Summary

Wrong Implementation of WARCSv1

1. There is no method to confirm Bob's credential.



Summary

Wrong Implementation of WARCSv1

1. There is no method to confirm Bob's credential.

2. Send OTP in clear text without encrypted tunnel.

Alice

!

Eve can read the OTP

Bob

Eve

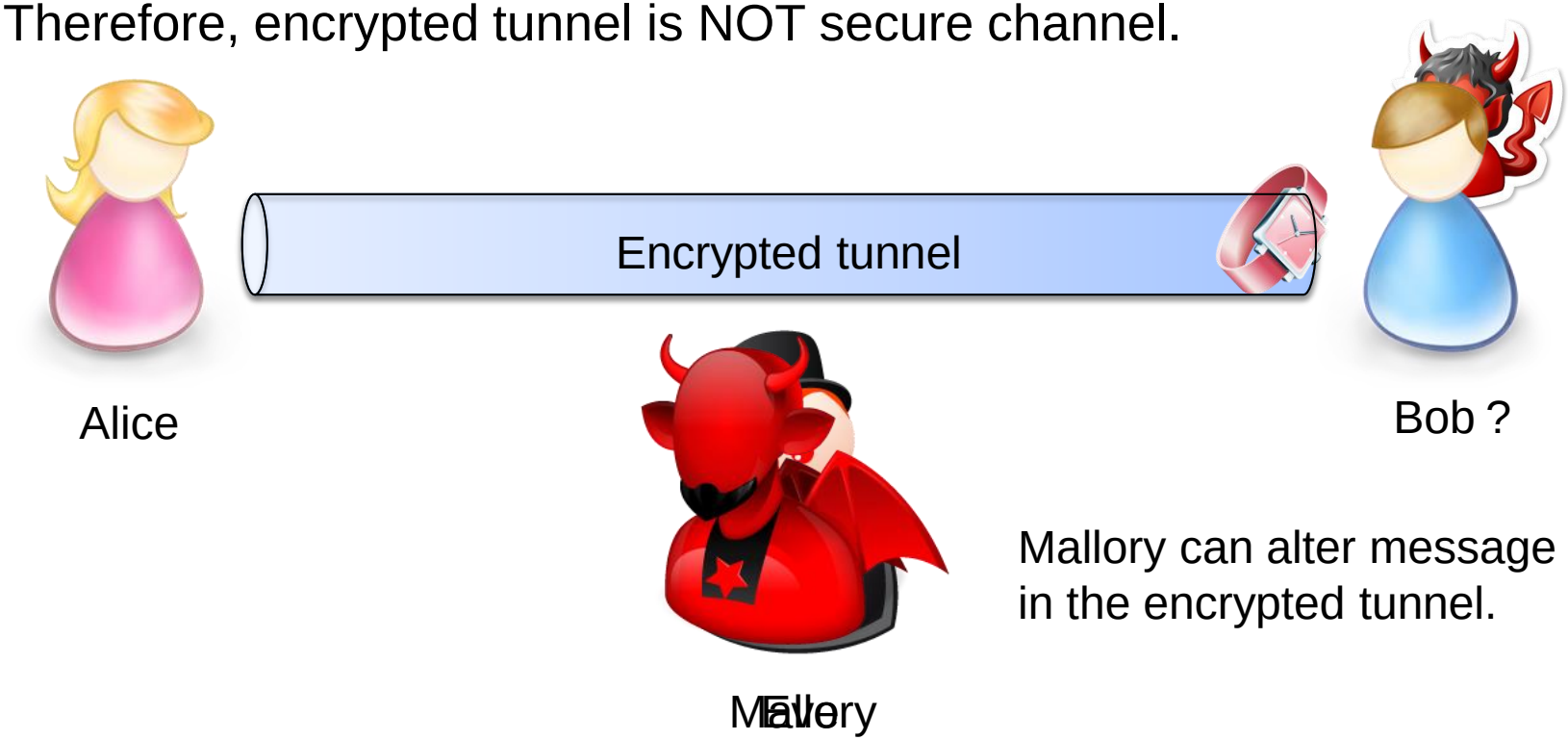
Summary

Wrong Implementation of WARCSv1

1. There is no method to confirm Bob's credential.

2. Send OTP in clear text without encrypted tunnel.

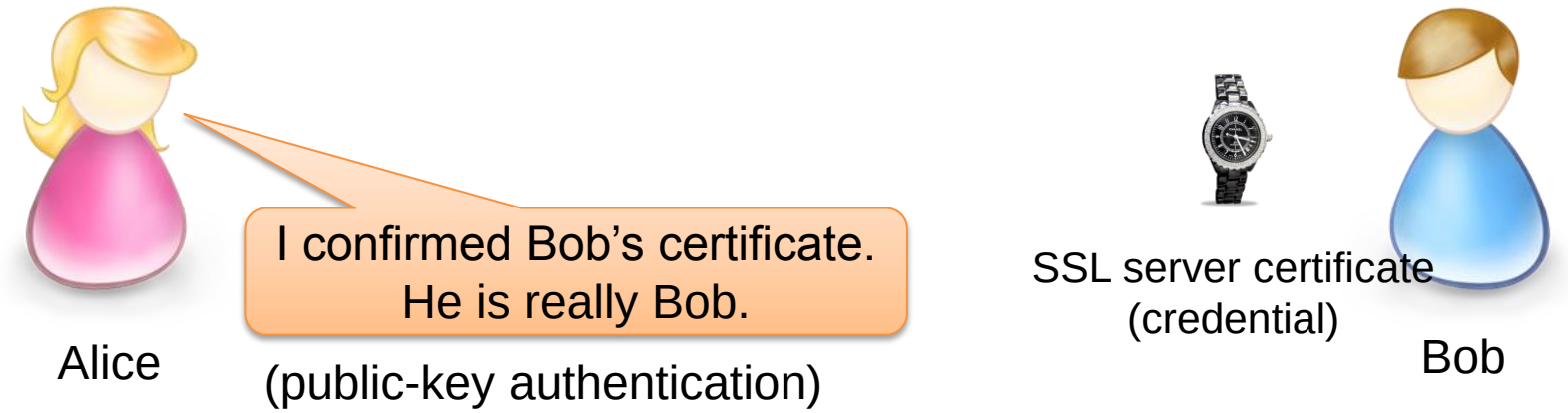
Therefore, encrypted tunnel is NOT secure channel.



Summary

Implementation of WARCSv2

1. Alice confirm Bob's credentials by the server certificate.



Summary

Implementation of WARCSv2

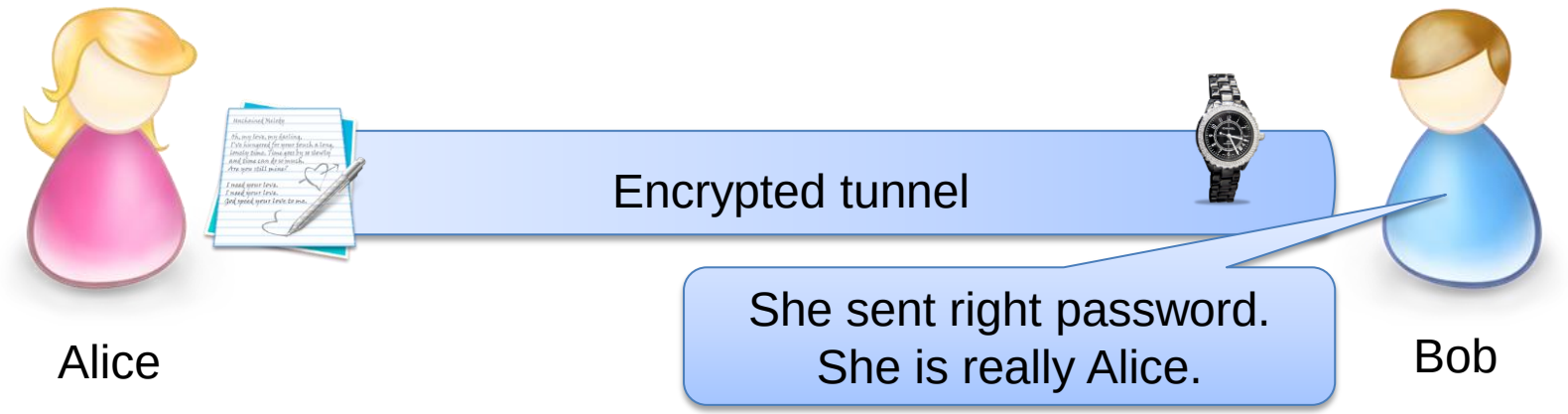
1. Alice confirm Bob's credentials by the server certificate.

2. Encrypted tunnel (SSL tunnel) between Alice and Bob is established.
   (at this moment, Alice's credential is not confirmed by Bob)

Summary

Implementation of WARCSv2

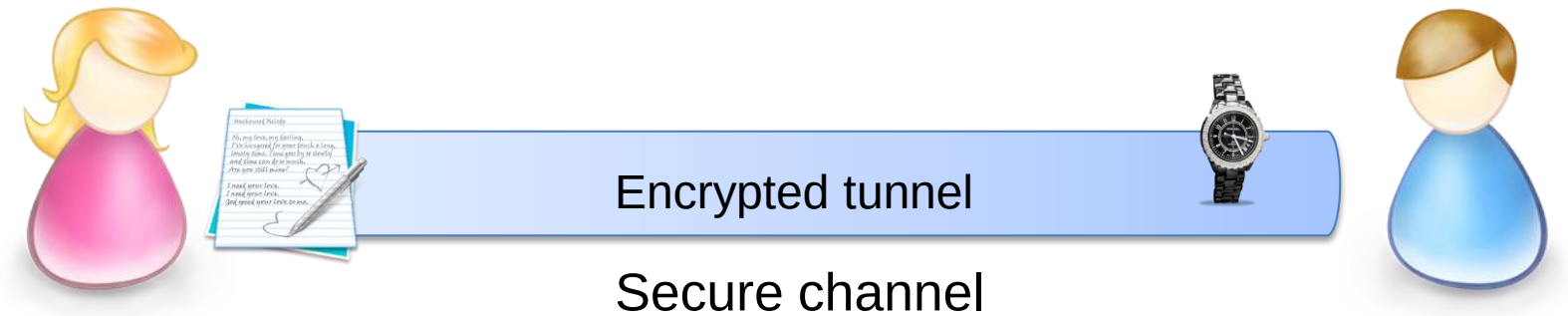
1. Alice confirm Bob's credentials by the server certificate.
2. Encrypted tunnel (SSL tunnel) between Alice and Bob is established.
   (at this moment, Alice's credential is not confirmed by Bob)
3. Alice send temporary password (credential) to Bob via encrypted tunnel.



Summary

Implementation of WARCSv2

1. Alice confirm Bob's credentials by the server certificate.
2. Encrypted tunnel (SSL tunnel) between Alice and Bob is established.
   (at this moment, Alice's credential is not confirmed by Bob)
3. Alice send temporary password (credential) to Bob via encrypted tunnel.
4. Secure channel is established. Let's start remote maintenance.



Alice

Expert

Bob

WARCS VPN gateway
and
Machine Network

Summary

- Overview of WARCS(v1)
  - WARCSv1 is developed in 2003. Recently, we faced on some problem.
  - Especially, vulnerability is found in the WARCSv1.
  - The vulnerability is come from wrong implementation of authentication.
- We develop new WARCS(v2)
  - We change the authentication procedure. The new procedure is based on standard SSL method.
  - WARCSv2 is installed SPring-8 and SACLA, and in operation.

- Again, what is wrong?
  - We did not pay careful attention to the authentication procedure.
  - What we (unprofessional in cryptography) think is may be wrong.

Main part of this work is done by my colleague, SAKAMOTO. He got a master's degree by this work in Sep. 2013.

Backup slides

Private-key sharing method of WARCSv1

- Maybe, public/private key pair are generated from one-time password without any random number?
  - This case is also vulnerable, because entropy of key-generation seed is too small.
  - The password length and character is public in the proceeding paper, only 9999999 case.
  - Attacker can generate hash table for all of the public/private key pair in advance.

Remaining Vulnerability in WARCSv2

- WARCSv2 is vulnerable to MITM exploit on the phone call
  - At this moment, this vulnerability is not resolved.

- We plan to apply two-factor authentication
  - We have Public-Key Infrastructure (PKI) system on the IT system.
  - By applying PKI to WARCSv2 additionally, attacker (who don’t have client certificate of the PKI system) can not authenticate.

OpenBlocks AX3

69,800 JPY (= 700 USD)

- ARMADA XP, 2core, 1.33GHz (ARM Cortex-A9)
- 1GB main memory
- 128MB NOR Flash ROM
- 1x SATA (for internal SSD, option)
- 1x eSATA (for external storage)
- 2x 1000BASE-T
- 2x USB 2.0
- 1x RS-232C
- 13.0W Power Consumption
- Debian GNU/Linux 6 and 7 (7 is officially supported by Debian community)

We also plan to use the OBAX3 as embedded device for accelerator control.