DIVERSID
AUTENTICACIÓN

OTP (One Time Password) based Mutual Authentication

Real-time generated OTPs
versus
stored OTPs comparison

www.diversid.com

Madrid, July 2007

Introduction

The goal of this comparison is to better understand the benefits provided by mutual authentication solutions such as the one based on the patent commercialized by Diversid, against other methods, based on real-time generated OTPs, some of them already in the market.

One of the simplest and safest ways of providing mutual authentication is done through OTPs exchange.

There are basically two ways of obtaining OTPs:

- By generating them at the time they are going to be used (Real-time)
- By generating them previously in a random process and storing them on a device with memory

In the first case OTPs are **generated** and displayed on the user device (such as a token), while in the second case OTPs are **stored** and displayed on the user device.

In the following slides we will demonstrate that the way OTPs are obtained has an impact on the authentication process, on the usability from the user perspective and on the resources consumption and therefore on operational costs.

To do the comparison we consider the real-time generated OTPs based on the challenge-response method.

Real-time generated OTPs
Characteristics

The mutual authentication process is supported by the following keys:

- Challenge X / Response X
- Challenge Y / Response Y

Challenges are random numbers obtained in real-time.

Responses are calculated with an algorithm that uses as parameters: the Challenge, a Value (k) shared by both parties and, for example, an event Counter.

Challenge-Response OTPs exchange between parties A and B is:

- A sends a random Challenge X to B
- B calculates the Response_X = f (K,Counter,X) and sends it to A with its own random Challenge Y
- A verifies the Response_X and, if it is correct, recognizes B party's authenticity and calculates the Response_Y = f(K,Counter,Y)
- B verifies the Response_Y and, if it is correct, recognizes A party's authenticity

Real-time generated OTPs
Keys exchange protocol

Customer - Token

PIN

1. PIN is entered in the "token" and both Challenge_X and the expected Response_X are displayed

6. Visual verification of the Answer

7. Enters PIN

8. Enters Challenge_Y

9. Response_Y is displayed

2. Challenge_X →

← 5. Response_X, Challenge_Y

10. Response_Y →

← 13. Accept / Reject

Server

3. Uses Challenge_X to calculate the Response_X

4. Sends the Response_X plus the new Challenge_Y

11. Validates the Response_Y

12. Accepts o rejects the Customer

Stored OTPs
Characteristics

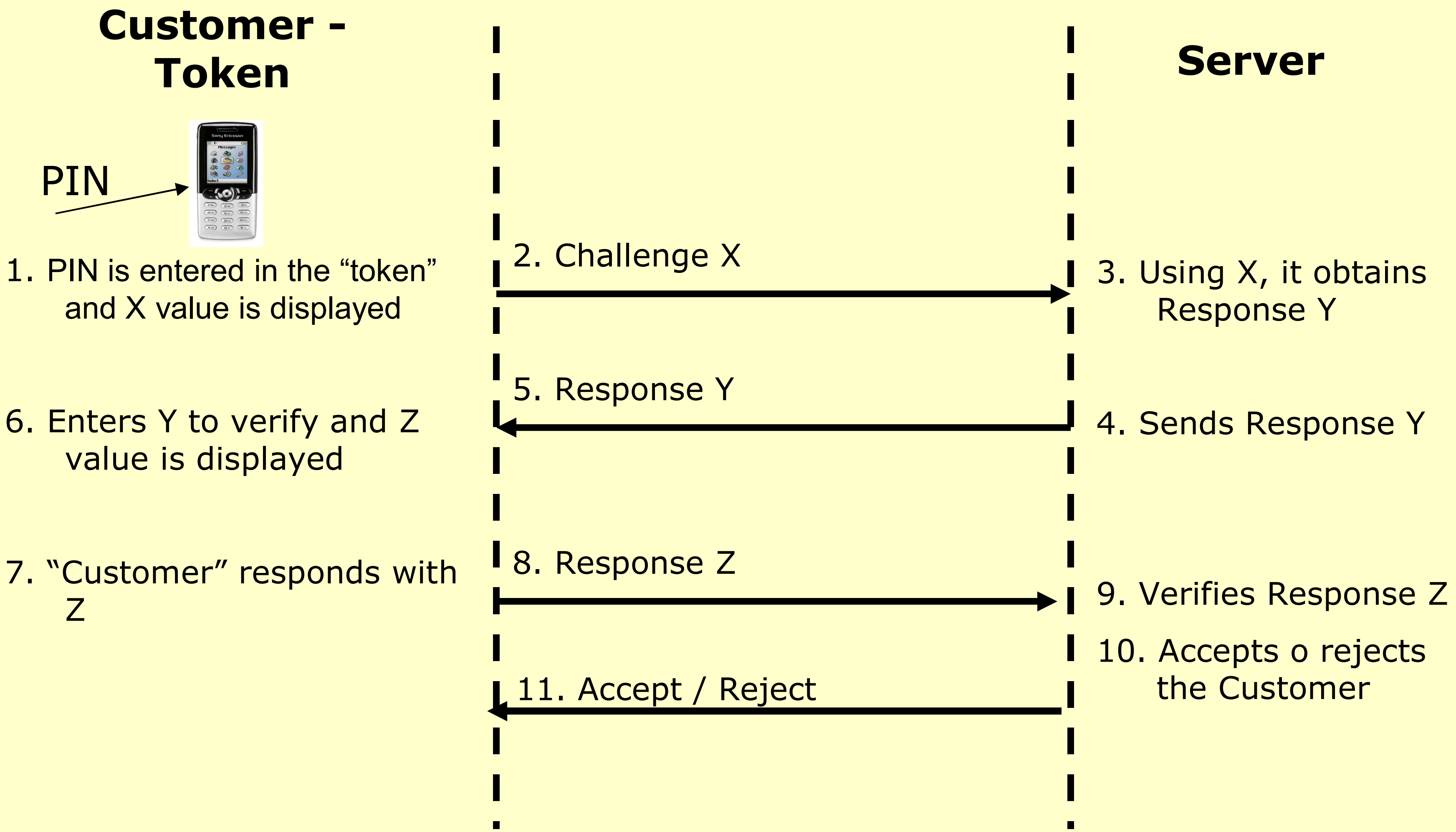
“Responses”: three OTP keys (X, Y, Z) obtained from a row in a Key Table, randomly generated from a previous process. This Table is stored in both the Token and the Server. Once a row has been used, it will not be displayed again.

Stored OTPs exchange between participants A and B is:

- A sends a random number X as Challenge for B
- B obtains Response Y accessing the same row as X and sends it to A
- A verifies Response Y and, if correct, recognizes B party’s authenticity and sends to B its Response Z, which is in the same row as X and Y
- B verifies Response Z and, if correct, recognizes A party’s authenticity

Stored OTPs

Keys exchange protocol



Customer - Token

Server

PIN

1. PIN is entered in the “token” and X value is displayed

2. Challenge X

3. Using X, it obtains Response Y

4. Sends Response Y

5. Response Y

6. Enters Y to verify and Z value is displayed

7. “Customer” responds with Z

8. Response Z

9. Verifies Response Z

10. Accepts o rejects the Customer

11. Accept / Reject

Real-time generated versus stored OTPs
Comparison from a resources consumption perspective

<u>In an authentication process with real-time generated OTPs protocol and algorithm,</u>

The User must:

- carry out two more additional steps than the ones carried out by stored OTPs (a visual verification and the operation of entering the Pin in the token.)
- resynchronize the token more frequently as a consequence from the fact that the counter changes every time the User requests a Challenge regardless whether it has been received by the Server or not.

At the Token you need:

- 3 additional data access step compared to the ones carried out by stored OTPs.
- 3 additional data update step compared to the ones carried out by stored OTPs.
- 3 instances of algorithm's application for calculating the Challenges and Responses, which are not needed in stored OTPs case.

At the Server you need:

- 3 additional data access step compared to the ones carried out by stored OTPs.
- 3 additional data update step compared to the ones carried out by stored OTPs.
- 3 instances of algorithm's application for calculating the Challenges and Responses, which are not needed in stored OTPs case.
- to carry out a greater number of synchronization operations for the token's counter, as a consequence from the fact that the counter changes every time the User requests a Challenge regardless whether it has been received by the Server or not.

Conclusions based on the detailed study of both processes that you may find at the end of the presentation as an Annex.

Summary
Operational requirements and costs comparison

- For real-time generated OTPs, the User must perform two more steps for the authentication
- For real-time generated OTPs, the Token increases the cost corresponding to a greater calculation capacity necessary to run the OTP calculation algorithm.
- For real-time generated OTPs, the Server requires a significantly greater process capacity to run the OTPs calculation algorithm. Due to the great operation volume it will need to perform per second, this will impact both the cost and the possibility of improving response times.
- For stored OTPs, the Token increases the cost corresponding to the necessary storage to hold the key table (approximately, 50K for 4000 authentication operations. If the token is a mobile phone this requirement will not be necessary, as there is an option for recharging the keys.)
- For stored OTPs, the Server increases its cost corresponding to the necessary storage to hold the key table (a memory of approximately 50 K for 4000 authentication operations per User, of which only 1 K would be necessary to hold with online access. A million Users would require 1GB. An average PC can have around 160 GB for data storage.)

Conclusions based on the detailed study ot both processes that you may find at the end of the presentation as an annex.

Summary

Performance comparison from a vulnerability perspective

Vulnerability	**Real-time generated OTPs**	**Stored OTPs**
Service denial	If a User delivers a Challenge (Q) to a fraudulent page that requests it, the Token's counter changes. If this is repeated a number of times greater than the synchronization margin (look-ahead parameter) the User will later have rejection problems when the Server tries to synchronize, producing a 'service denial'.	This cannot happen at stored OTPs Token, as it does not mark the row as used until the correct K2 is introduced and K3 displayed.
Fraudulent service degradation	Each time a Q challenge is received by the Server, it must recalculate it and, if it does not match, it must also calculate the ones corresponding to the admitted error margin. From this fact, PCU resource consumption issues might appear which, in turn, could produce 'service degradation' in the event of a massive attack.	In the same situation, stored OTPs would only carry out one I/O Operation (all key rows are in the same I/O page), which does not imply a consumption that could provoke 'service degradation'.
Identity impersonation (Phishing, Pharming , etc)	It is not possible, as the Server does not authenticate the User until it receives the Response and this cannot be known.	It is not possible, as the Server does not authenticate the User until it receives the Response and this cannot be known.
Man in the middle	It seems feasible to implement a solution that avoids this issue.	Some of the Diversid applications address and avoid MITM.

DIVERSID
AUTENTICACIÓN

ANNEX

Real-time generated OTPs versus stored OTPs comparison

www.diversid.com

Madrid, July 2007

Comparative study from an operational perspective (1/2)

STEP	A: PROTOCOLO OTP Generada	B: PROTOCOLO OTP Almacenada	ESTIMATED DIFFERENCES
0	The User enters Pin in the Token.	The User enters Pin in the Token.	Similar operation
1	The Token generates a User Challenge X (random number) and calculates the corresponding R(X) Response, using the K value and the counter shared with the Server, and both are displayed on the Token's screen.	The Token accesses X value from the first non-used key package, and displays it on the Token's screen.	**Operation for case A Token**: It accesses the counter It updates the counter It accesses the K value It generates X Challenge It calculates R(X) Response **Operation for case B Token**: It accesses X from the first non-used row
2	The User enters X to be sent to the Server in the PC.	The User enters X to be sent to the Server in the PC.	Similar operation
3	The Server receives X, calculates the corresponding R(X) Response using the K value and the counter that it shares with the User and then generates a Y Challenge (random number) which it stores.	The Server receives X, it checks that the first non-used row has the received X as first key and, if it matches, marks the row as used.	**Operation for case A Server**: It accesses the counter It updates the counter It accesses the K value It calculates R(X)Response It generates Y Challenge It stores Y **Operation for case B Server**: It accesses the keys from the first non-used row It checks that it has the received X as first key It updates the row marking it as used.
4	The Server sends the R(X) to the PC along with the Server's Y Challenge.	The Server sends the Y key that is in the same package as the received X to the PC.	Similar operation

Comparative study from an operational perspective (2/2)

STEP	A: Real-time generated OTPs Protocol	B: Stored OTPs protocol	ESTIMATED DIFFERENCES
5	The PC displays R(X) to the User and the new Y Challenge.	The PC displays Y to the User.	Similar operation.
6	The User visually verifies that the R(X) Response received in the PC matches the one shown by the Token as expected.		**Not needed in B case**
7	The User enters the Pin again in the Token.		**Not needed in B case**
8	The User enters in the Token the Y Challenge received in the PC.	The User enters in the Token the Y received in the PC.	Similar operation.
9	The Token calculates the corresponding R(Y) Response using the K value and counter that it shares with the Server and displays it on the screen.	The Token checks that the Y entered matches the one in the same row as the X sent and, if it does, it displays the Z on the screen, which must be used next.	**Operation for case A Token**: It accesses the K value It accesses the counter It updates the counter It calculates the R(Y) Response **Operation for case B Token**: It checks that the Y entered matches the Y from the row being used
10	The User enters the R(Y) Response to be sent to the Server in the PC.	The User enters the Z Response to be sent to the Server in the PC.	Similar operation.
11	The Server receives the R(Y) Response from The User, and it calculates which one should arrive for the Y value stored when it was generated (step 3) , if the calculated one matches the received one, it validates the Response closing the mutual authentication process.	The Server receives the Z Response from the User, and it checks that it matches with the one from the same row as the Y sent; if it does, the Server validates the Response closing the mutual authentication process.	**Operation for case A Server**: It accesses the K value It accesses the counter It updates the counter It accesses the Y value stored in step 3 It calculates R(Y) It checks that both R(Y) match **Operation for case B Server**: It checks that the Z received matches the Z from the row being used
12	The Server, depending on the result, accepts or denies the access.	The Server, depending on the result, accepts or denies the access.	Similar operation.
13	The Server sends the mutual authentication result to the PC.	The Server sends the mutual authentication result to the PC.	Similar operation.