ActivCard Synchronous Authentication

This paper describes ActivCard authentication based on the Data Encryption Standard (DES) algorithm. It includes technical definitions of the DES algorithm and the following security terms and technologies: (1) ActivCard asynchronous (challenge/response) authentication; (2) ActivCard's patented synchronous authentication (using three variables of time, event, and secret key); (3) encryption based on the sharing of secret keys; and (4) selected ANSI communications security standards. Where appropriate, diagrams are included to illustrate ActivCard methodology. ActivCard core encryption technology can be used in many different applications – not just applications requiring user authentication (i.e. host authentication, message authentication code generation, encryption of electronic mail messages, and smart-card-based financial transactions).

The DES Algorithm

The DES algorithm is a publicly available algorithm that has been extensively studied and tested since its publication almost three decades ago. It is one of the most well-known and widely used secret-key cryptosystems in the world and is therefore considered one of the most reliable. It is for this reason that ActivCard employs the DES algorithm in its core technology. The National Institute of Standards and Technology (NIST) has certified DES; *and it is re-certified every five years*. Certified for both authentication and financial transactions, DES is considered by industry experts to be the safest encryption algorithm available – because: (1) it is standards-based (non-proprietary); (2) it has been under serious scrutiny since 1979; and (3) it uses a 56-bit key (triple DES uses a 112-bit key).

ANSI and ISO Standards Compliance

ANSI is an administrator and coordinator of the U.S. private-sector, voluntary standardization system. The Institute represents the interests of approximately 1,400 domestic and international members from private organizations, corporations, and government agencies. ANSI does not itself develop American National Standards; rather it facilitates development by establishing consensus among qualified groups. ANSI promotes the use of U.S. standards internationally, advocates U.S. policy and technical positions in international and regional standards organizations, and encourages the adoption of international standards as national U. S. standards (when deemed appropriate).

ANSI is the sole U.S. representative to the International Organization for Standardization (ISO), one of five permanent members to the governing ISO Council, and one of four permanent members of ISO's Technical Management Board. U.S. participation, through ANSI's U.S. National Committee (USNC), is equally strong in the IEC. Through ANSI, the United States has immediate access to ISO and IEC standards development processes. In many instances, U.S. standards are taken forward through ANSI or its USNC to the ISO or IEC, where they are adopted in whole or in part as international standards.

The DES algorithm complies with the American National Standards Institute (ANSI) X3.92 standard for encryption.

ActivCard core encryption methodology complies with the following ANSI standards:

- ANSI X3.92 for encryption algorithms;
- ANSI X9.17 for secret key management;
- ANSI X9.9 for display of dynamic passwords;
- ANSI X9.24 for DES key derivation; and
- ANSI X9.26 for challenge/response, sign-on authentication.

ActivCard's Use of Encryption

Let's begin by defining "encryption" – the process that makes safe, confidential exchange of information possible in today's world of electronic communications. Encryption – the process of turning readable text into cipher text – ensures that data in transit can only be read by the intended recipient (whether a person or computer). Encryption disguises the message (for example, a password for user authentication) making it unintelligible to anyone (or any electronic device) other than the intended recipient.

Encryption utilizes a mathematical algorithm (a formula), such as DES, and a digital key (a series of bits) to encode a message at one end of a transmission and then decode it on the other end of the transmission. This digital "key" is a numeric value *that is part of the algorithm* for encrypting the text; and this key must remain confidential. ActivCard technology employs "secret" keys in a process called symmetric encryption – a method which makes use of a single secret key (called a constant) that is used on both sides of the exchange of information.

A "secret" key is a single key known only to the "participants" exchanging data. This secret key is used in the formula that encrypts and decrypts the data. Without this key, you cannot decrypt the data after it is exchanged. This secret key is used along with the DES algorithm to encrypt a challenge issued by either of the two parties involved in the exchange of information (or a third party). There are two kinds of "challenges" that can be issued in an ActivCard encryption session: external and internal.

'External Challenge' Defined

External challenges are used in challenge/response (asynchronous) encryption sessions. They can be generated by either of the two parties involved (for example, a client and an authentication server) or a third, independent party. (See *Figure 1.*) This external challenge is then used along with a secret key and the DES algorithm to create a "response" to the external challenge. The following holds true, no matter what the application involved:

- Both parties use a common challenge (which varies with each session);
- Both parties share a secret key – which is therefore "constant"; and
- Both parties use the DES algorithm, the "constant" secret key, and the "variable" challenge to generate an encrypted "response".

“External” Challenge Is Used by Both Parties

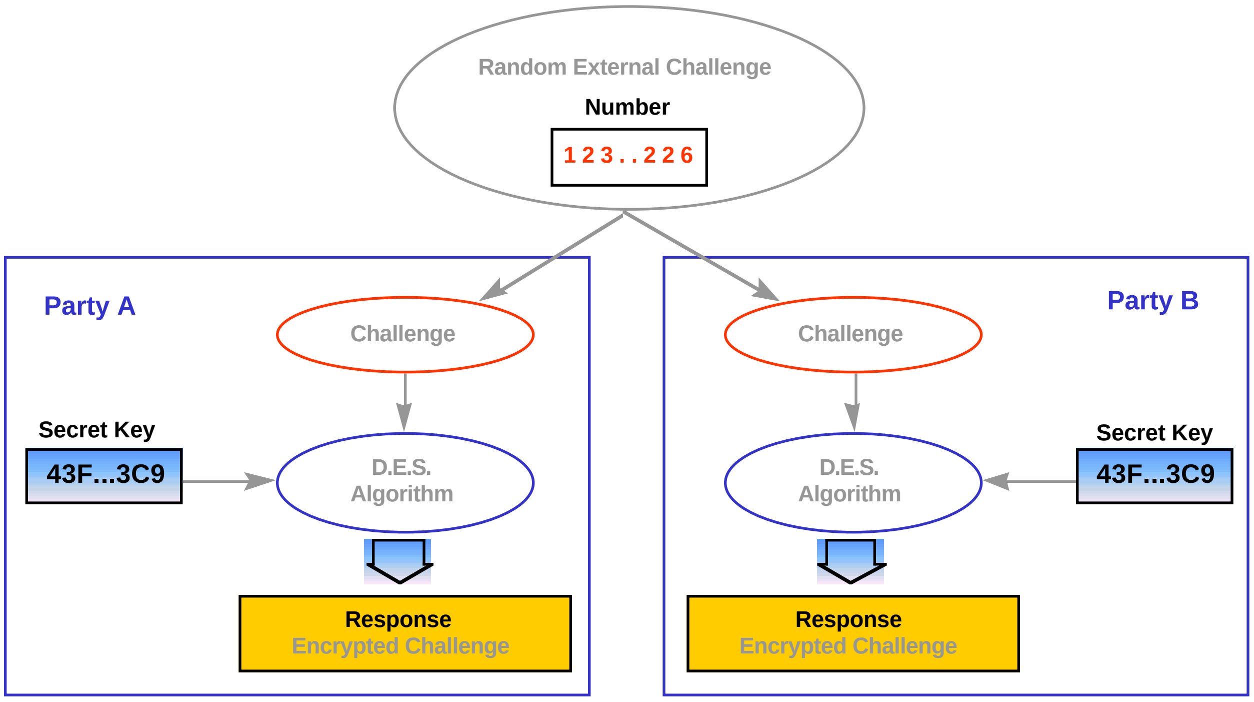


Figure 1: **Challenge/response employs an “external” challenge during the encryption process. To calculate the same response, both parties must share the same secret key (constant) and receive the same external challenge (variable). This challenge can be generated by either of the parties (or a third, independent party).**

‘Internal Challenge’ Defined

An “internal” challenge is generated by both of the two parties involved in the encryption session. It must be generated using the same variables, or else the two parties will not be able to conclude the encryption session successfully. (*See Figure 2.*) ActivCard uses an internal challenge in its synchronous mode of operation. Two variables are used to generate the internal challenge; they are derived from an internal event counter and an internal clock counter. This internal challenge, along with the shared secret key and the DES algorithm, generate the “encrypted response”.

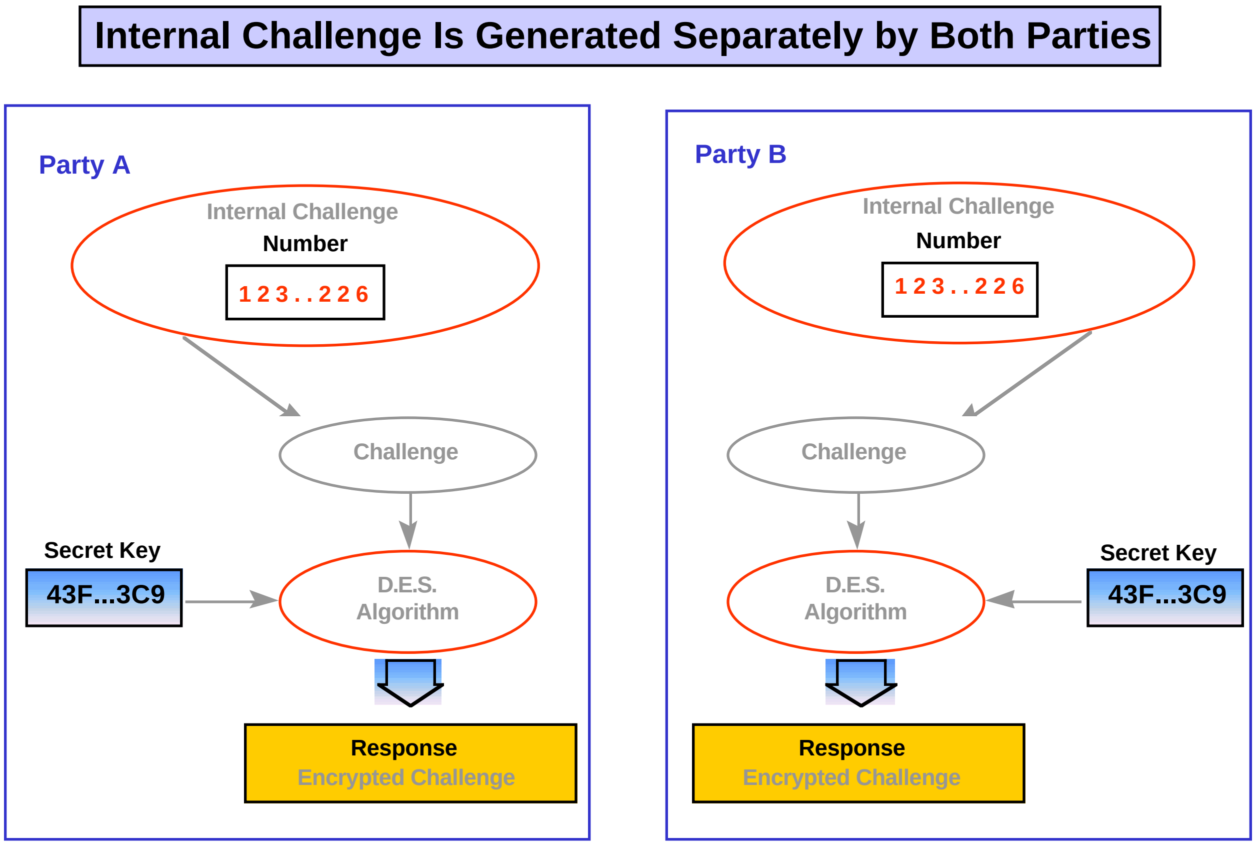


Figure 2: **Synchronous encryption sessions involve the use of an “internal” challenge, which is generated by both parties separately from one another. In order for synchronous encryption to succeed, the variables used by both parties must be “in sync”.**

ActivCard DES-Based Authentication: Overview

ActivCard user authentication is based on the exchange of encrypted secret keys, which are shared by two parties (a client – a token or smart card – and an authentication server). There are two modes of ActivCard authentication which make use of this encryption process and exchange of secret keys. One is commonly called “challenge/response” or *asynchronous* authentication. The second is synchronous authentication. From this point forward, this paper will cite user authentication to restricted network resources as the *task at hand*; however, please remember that ActivCard encryption methodology may be integrated into a variety of applications other than network user authentication, and this can be accomplished in diverse networking and communications environments.

Using the DES algorithm and secret keys, ActivCard tokens, smart cards, and authentication servers generate passwords which are exchanged during either asynchronous or synchronous encryption sessions. First, let’s define asynchronous as it relates to ActivCard core technology.

Asynchronous Authentication

ActivCard asynchronous mode is an authentication process whereby data (a password) is securely exchanged between a client (i.e. a token or smart card) and an authentication server. The method utilizes two variables – one constant and one that changes randomly (an external challenge). Both client and server hold the shared secret key (the constant); and both client and server use the DES algorithm and this secret key to encrypt a randomly issued, numeric challenge. The server issues the challenge; the client uses the challenge to create a response (or one-time-use password). The server uses the same two variables (the random challenge and the shared secret key) to generate the same response. This method is commonly referred to as challenge/response user authentication.

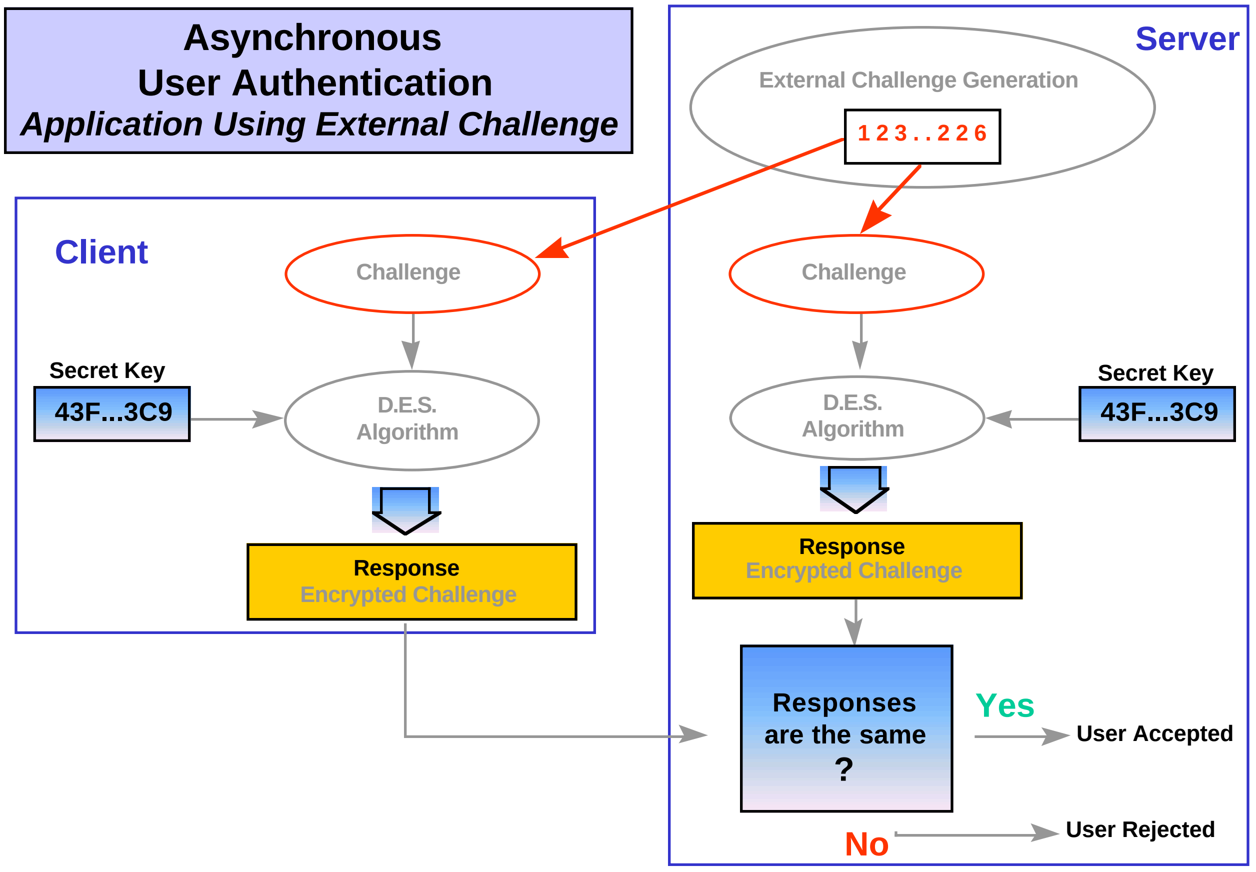


Figure 3: **ActivCard Asynchronous Authentication**

Asynchronous Authentication: Details

The following is a description of the asynchronous authentication process between a client (token or smart card) and an ActivCard authentication server. (See *Figure 3*.)

- The ActivCard server generates a *random* external challenge (variable number one) that is used as one factor in the encryption process on both server and client sides of authentication.

- Both server and client encrypt this random challenge using the DES algorithm and the specific DES secret key (variable number two) assigned to that client. This key remains constant.

- The encryption process yields a unique, one-time-use response – or password. This password is generated by both client and server. The server compares the passwords in order to confirm the identity (or "authenticate") the user to the server (or the server to the user).

In ActivCard authentication, every client (token or smart card) has a unique secret key assigned to it; this key is originally generated when the tokens and smart cards are initialized by the organization's security staff. These keys are not seen by the staff and can't be accessed by anyone – inside or outside of the organization.

Synchronous Authentication

Synchronous authentication can be defined as a methodology that is congruous (or consistent) on both client and server sides of the process. During ActivCard synchronous authentication, the client generates an internal challenge using values from its event counter and clock counter. The clock counter value is derived from the client's internal time clock. The client then encrypts the internal challenge using the DES algorithm and the derived secret key specific to that client. The exact same process takes place on the server side. However, the client must attach and send two "special" digits from its event and clock counters along with the encrypted challenge to the server, so that the server may stay synchronized with the client.

How does this technology work in a real applications?

Synchronous Authentication: Details

The following steps delineate synchronous user authentication using an ActivCard client (token or smart card) and ActivCard software. This password generation is based on three variables – a synchronous technology that is patented by ActivCard. (See *Figure 4*.)

The first series of tasks is performed by the client:

1. The client builds an internal challenge (independently of the server) using two variables (the client clock counter value *and* the client event counter value).

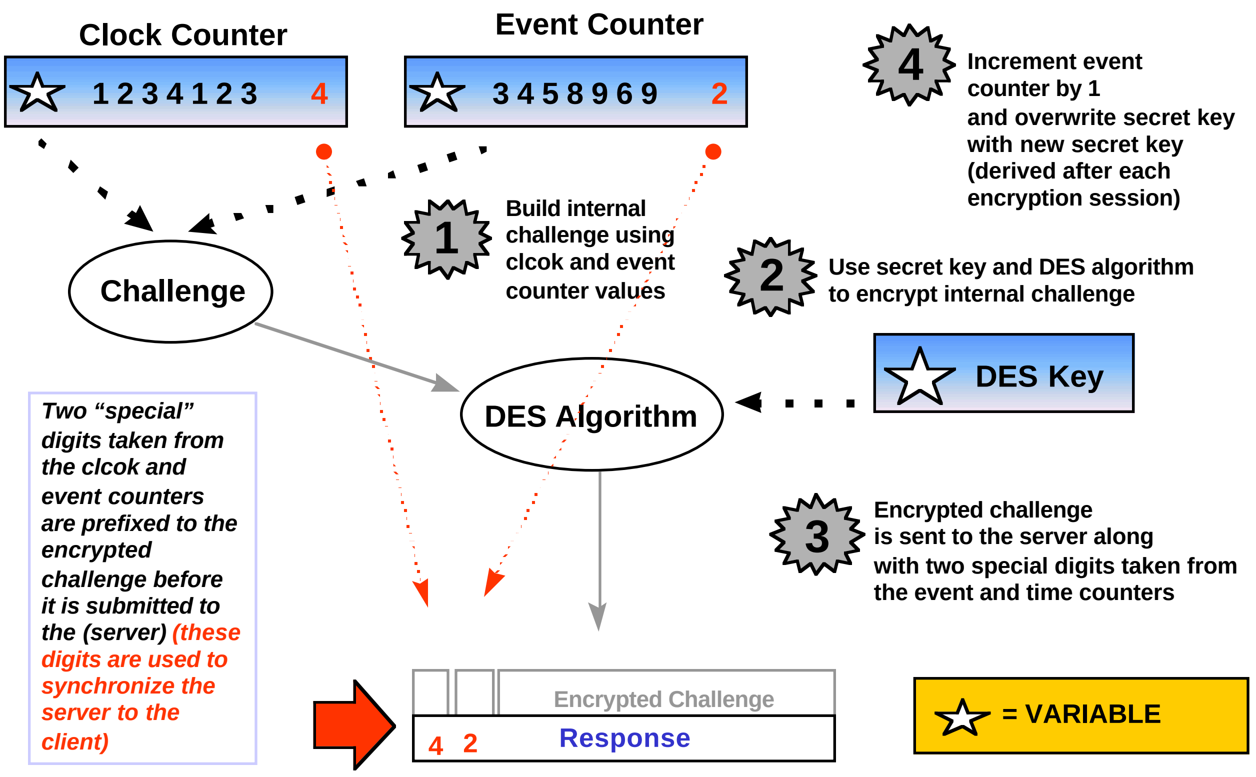
2. The client encrypts this internal challenge with the DES algorithm (which utilizes in its mathematical formula a *third* variable – a derived secret key *that is unique to that specific client and is used ONLY for that specific encryption session.*)

3. The client selects the two least significant digits (one each from the event and clock counters) and pre-fixes them to the encrypted challenge. The encrypted challenge and special digits together form a one-time-use password, which is sent to the server during a request for authentication to restricted resources.

4. The client increments its event counter by one and derives a new secret key, which overwrites the secret just used in the above encryption session. (Again, please refer to *Figure 4* for an illustration of "client-side" password generation based on three variables.)

This last step number four (secret key derivation) requires a short explanation before moving on to describe the server side of synchronous authentication. Each time an encryption session takes place, a secret key derivation algorithm (following the ANSI X9.24 standard) uses the event counter and the previous secret key to generate a new key for the next session. This derived secret key is the third variable, which makes ActivCard synchronous encryption one of the strongest security technologies available. (See *Figure 5*.)

Three-Variable Encryption Process



Clock Counter
1 2 3 4 1 2 3 4
Event Counter
3 4 5 8 9 6 9 2
4
Increment event counter by 1 and overwrite secret key with new secret key (derived after each encryption session)
1
Build internal challenge using clcok and event counter values
2
Use secret key and DES algorithm to encrypt internal challenge
Challenge
DES Key
DES Algorithm
Two "special" digits taken from the clcok and event counters are prefixed to the encrypted challenge before it is submitted to the (server) (these digits are used to synchronize the server to the client)
3
Encrypted challenge is sent to the server along with two special digits taken from the event and time counters
Encrypted Challenge
4 2 Response
= VARIABLE

Figure 4: ActivCard synchronous authentication from the *client side of the process*. Refer to Figure 6 for an illustration of an application using these two special digits to synchronize client/server internal challenges.

Secret Key Derivation

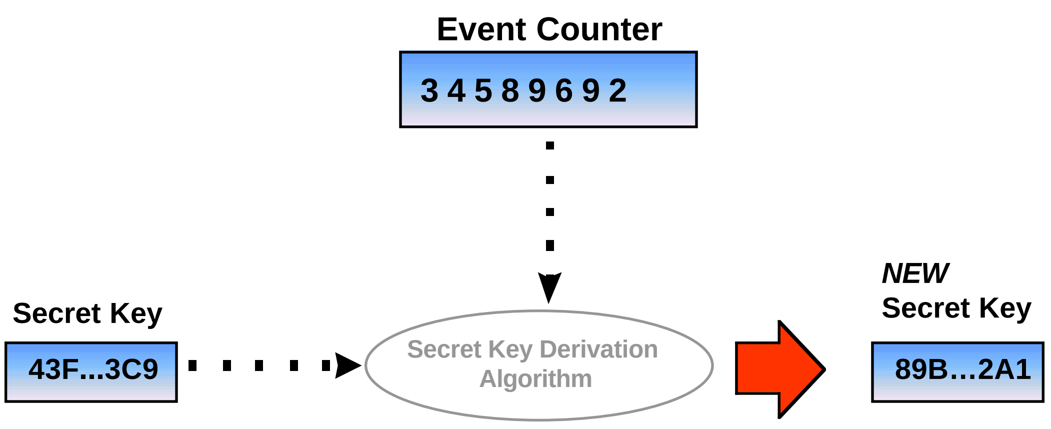


***Figure 5:* During ActivCard synchronous authentication a new secret key is derived after each encryption session.**

The second series of tasks in synchronous authentication is performed by the server, which authenticates the user based upon a password match. Because the server and client must do exactly the same calculations with the same three variables in order to compare meaningful results at the end of the session, the server variables must be “in sync” with the client variables at all times.

Remember that when the encrypted internal challenge is sent to the server, *two special digits are prefixed to it, and sent along with it.* (See *Figure 6*.) Recall that these digits are the least significant digits from the client’s clock and event counters and are used by the server to synchronize itself to the client – *prior* to performing the identical steps on its side. Every time a user requests authentication via a synchronous password, the server takes specific steps to determine if it is synchronized with the client.

1. First, the server looks at the two digits that are prefixed to the password to see if the client digits match the server digits.

2. Second, *if required,* the server re-synchronizes its event and clock counters to match the client’s (within defined security parameters) and then derives the necessary number of secret keys to “catch up” to the client and have the same secret key as the client has for itself. (Please see following sections on re-synchronization describing in detail how this is accomplished.)

3. When the server has determined that its digits match (or are re-synchronized), it builds its own internal challenge using its clock and event counter values, encrypts that challenge using the appropriate secret key and the DES algorithm, and compares the client’s and server’s encrypted challenges to determine if access should be permitted.

4. *Only* when the match is successful, does the server increment its event counter by one and derive a new secret key for that client.

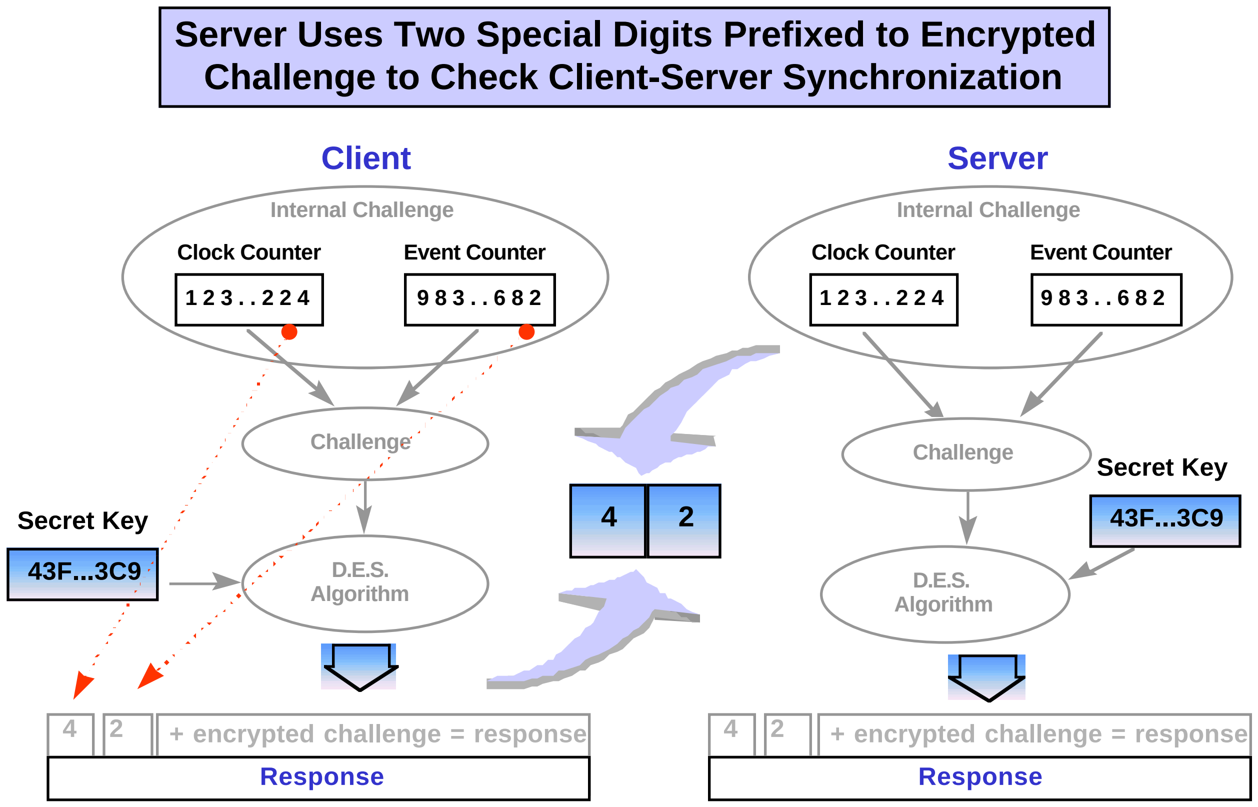


Figure 6: **The two special digits prefixed to the client's encrypted challenge are the least significant digits from its clock and event counters. The server reads these variable digits (and re-synchronizes if necessary) with its own event and clock variables – prior to generating a *single* encrypted challenge of its own.**

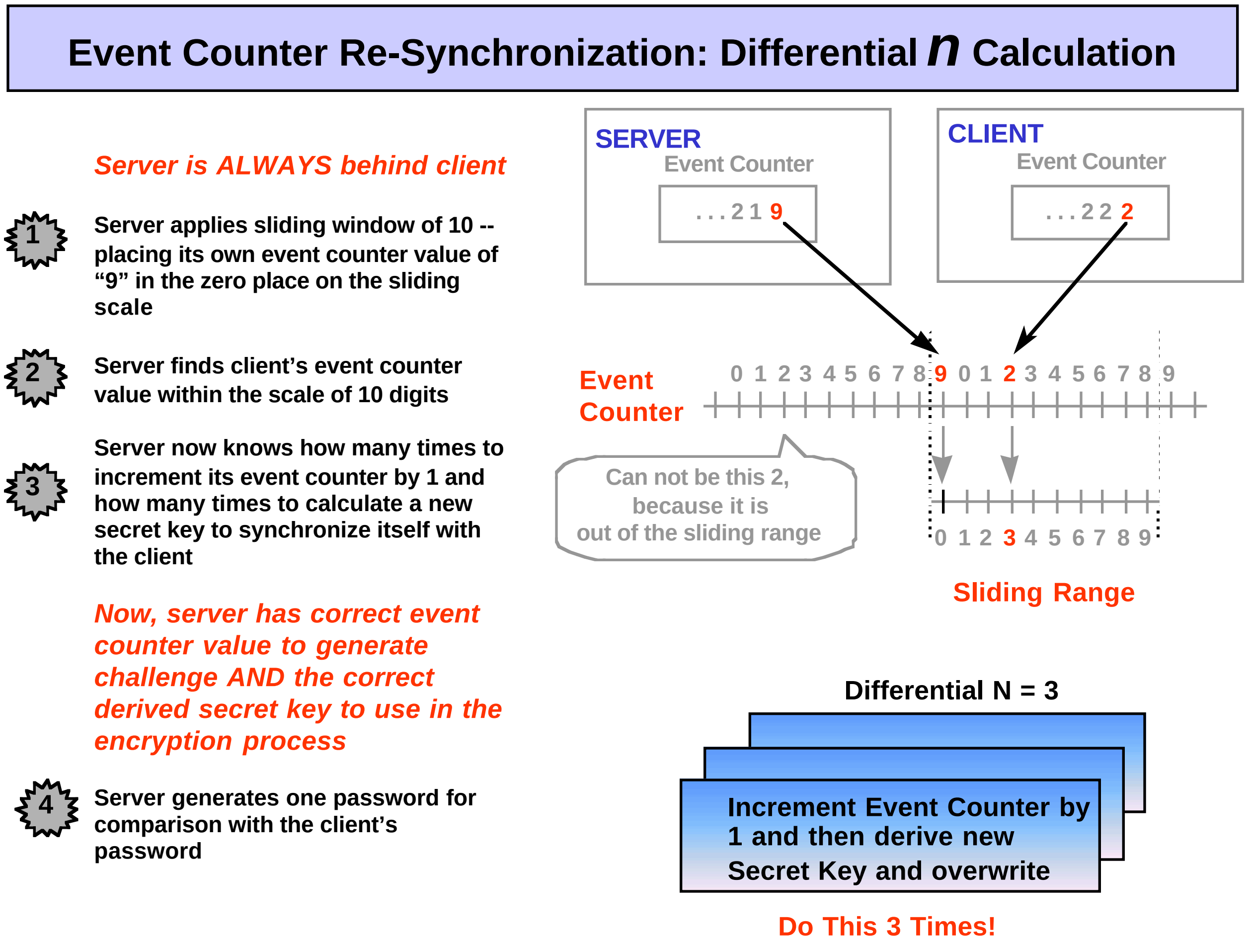
Re-Synchronizing Event Counters

How is it possible for the client and server become out of sync with each other? What does the server do when its event and/or clock counter digits do not match the digits sent by the client? The following section describes how the event counters may become out of sync with one another, and how the server re-synchronizes the event counters.

The client increments its event counter and derives a new secret key every time it goes through an encryption session. The client event counter tracks of the number of times the client has generated encrypted responses to its internal challenges – whether submitted to the authentication server for access to a particular resource, or not. Each time the client encrypts an internal challenge using the DES algorithm and the secret key, it increments its own event counter by one and then derives a new secret key and overwrites the one it just finished using. This "new" secret key derivation takes place at every encryption session; remember that the secret key is one of the variables used in the ActivCard three-variable, patented synchronous process.

On the other hand, the server increments its event counter and derives a new secret key for the client *only after* a successful match of passwords. If a client generates the internal challenge and creates the encrypted response, *but the response is not sent to the authentication server*, the client will – nevertheless – increment its own event counter by one, and the client will derive a new secret key for itself. However, now those values will not match the ones in the server, *which will have done nothing.*

How does the server re-synchronize its event counter to the client event counter during the ensuing encryption session? It uses one of those two special digits (the least significant digits from the client's event and clock counters) that are sent along as part of the password during each request for authentication. (See *Figure6)* The server always assumes its event counter is *behind* the client's event counter. Why? The server does nothing, unless a user requests access via a password generated by a client; therefore, the server's event counter *can never be ahead of the client*. When the server determines that the event counters do not match, the server uses a "sliding window" of 10 digits to re-synchronize its event counter to that of the client. This range of 10 digits (0-9) forms the "sliding window" of 10 – which can be applied in a re-synchronization formula to calculate the differential between the two clock counter values. (See *Figure 7*.)



Event Counter Re-Synchronization: Differential *n* Calculation

Server is ALWAYS behind client

1. Server applies sliding window of 10 -- placing its own event counter value of "9" in the zero place on the sliding scale
2. Server finds client's event counter value within the scale of 10 digits
3. Server now knows how many times to increment its event counter by 1 and how many times to calculate a new secret key to synchronize itself with the client

Now, server has correct event counter value to generate challenge AND the correct derived secret key to use in the encryption process

4. Server generates one password for comparison with the client's password

Figure 7: **The server calculates quickly and easily the differential between its event counter and the client event counter. It then increments its event counter and derives a new key three times in a row, before generating an internal challenge for the DES encryption process.**

Re-Synchronizing Clock Counters

A clock counter is a sequential counter which is incremented by one digit at regular intervals. The "one" represents a unit of time. The unit of time might be one hour, 15 minutes, 2 minutes, or one-half second. ActivCard clock counters in both client and server are derived from internal time clocks which move forward at half-second increments. The client has an internal clock, from which it derives its own clock counter value; and the server has its internal clock, from which it derives its clock counter value. Each time the client sends a password to the server, it sends along with it the least significant digit of its clock counter, and the server uses this single digit to re-synchronize itself to the client before it begins calculating a password for authentication. (See *Figure 8.*)

The following steps occur during the re-synchronization of the server clock counter value to the client clock counter value:

1. The server receives the password which includes the client clock value (the least significant digit) along with the event counter value digit and the encrypted internal challenge.

2. The server compares its clock value digit to the client clock value digit. Because the server is now dealing with time, it cannot "know" if its own clock value is ahead or behind the client clock value. There may have been some "drift" in time between the two clocks, after the internal clocks were originally set. It could be several seconds (or minutes) ahead or behind. However, ActivCard patented technology easily discerns – through the use of only these two, single digits – not only the differential in time, but also whether or not the server is ahead or behind, and whether or not it should re-synchronize (within security parameters).

3. How does the server do this? The server applies the same concept as it applied when correcting for an event synchronization differential. It uses a sliding window of 10. Because there are 10 single digits in the sliding "scale", there must be four digits on one side of the anchor point (zero) and five digits on the other side of the anchor point. *ActivCard has arbitrarily set the sliding scale to minus five (-5) and plus four (+4). (See Figure 8.)*

4. Please follow along with *Figure 8* during this step. The server applies the sliding scale to its clock counter, anchoring the scale's zero place at the location of its least significant digit (2). The least significant digit of the client's clock counter (4) then falls somewhere within the 10-digit range of the sliding scale. The server simply locates the client's digit in relation to its own least significant digit; then it looks directly opposite to the sliding scale and locates ***N*** (in *Fig. 8* that value is +2). The server then re-synchronizes its clock value by that amount. (See *Figure 9* on the following page for a second diagram illustrating an example where ***N*** is a negative (rather than positive) number. It does not matter to the server if its clock is ahead or behind the client. The sliding scale will always result in a correct re-synchronization – within given security parameters set by the organization's IS staff.)

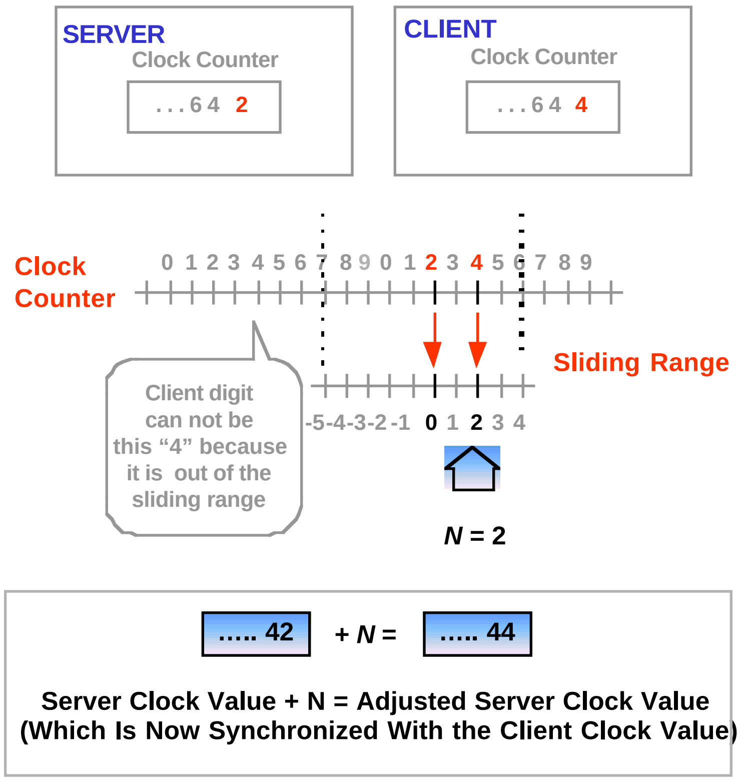
Clock Counter Re-Synchronization: Differential *n* Calculation

Server does not know if it is behind or ahead of the client

1. Server applies sliding window of 10, placing its own clock counter digit of “2” in the zero place on the sliding scale
2. Server finds client’s clock counter digit (4) within the sliding scale of 10
3. Server calculates the differential N between server and client digits

Now, server has correct clock counter value in order to generate challenge

4. Server generates one password for comparison with the client’s password



Server Clock Value + N = Adjusted Server Clock Value
(Which Is Now Synchronized With the Client Clock Value)

Figure 8: The server calculates quickly and easily the differential between its own clock counter value and the client clock counter value. It then re-synchronizes its clock counter before generating an internal challenge for the DES encryption process.

Clock Counter Re-Synchronization: *Example No. 2*

Server does not know if it is behind or ahead of the client

1. Server applies sliding window of 10 -- placing its own clock counter digit of “2” in the zero place on the sliding scale
2. Server finds client’s clock counter digit (9) within the sliding scale of 10
3. Server calculates the differential N between server and client digits

Now, server has correct clock counter value in order to generate challenge

4. Server generates one password for comparison with the client’s password

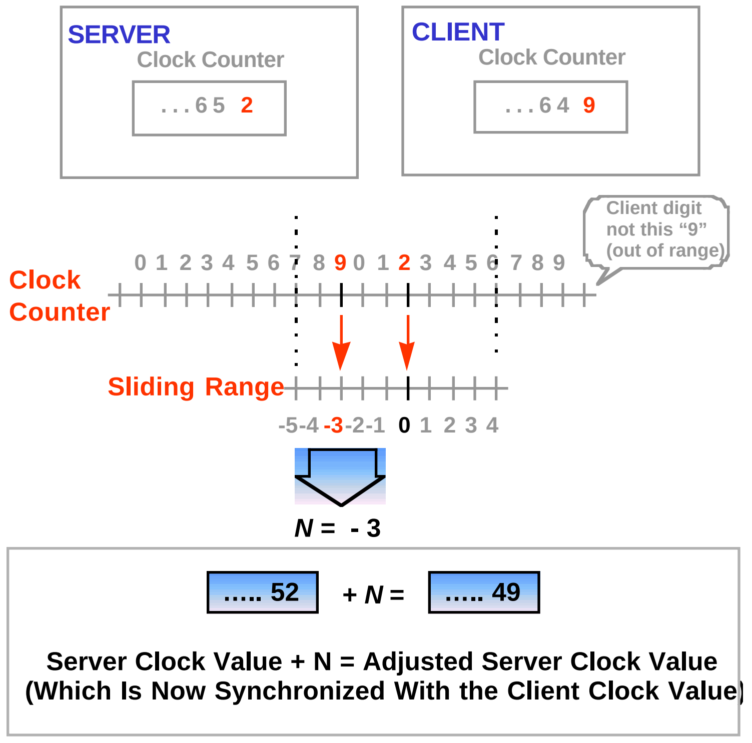


Figure 9: **The server calculates quickly and easily the differential between its own clock counter value and the client clock counter value. It then re-synchronizes its clock counter before generating an internal challenge for the DES encryption process.**

Summary

Most organizations today rely on disparate groups and technologies to manage mainframes, host servers, client systems, and an abundance of software applications required for local and wide area networking, internetworking, and organization-wide access through the Internet to World Wide Web. This complex mix of operating systems, computer products, platforms, and applications makes efficient (and complete) communications security a difficult puzzle, if not a nightmare to solve. It is common for organizations to mix security solutions from multiple vendors; these products often work on different operating systems and utilize a variety of databases in a complex overall operating environment. ActivCard core encryption technology can be used in many different applications – not just applications requiring user authentication. ActivCard encryption technology is used in host authentication, user authentication and message authentication. It can be used to encrypt electronic mail messages, and protect financial, smart-card-based transactions. Because it is standards-compliant and based on an open architecture, it is possible to integrate ActivCard core encryption technology across the entire, complex mix of systems, computers and platforms – making the communications puzzle a little easier to solve.