STUDI ALGORITMA PROTOKOL OTENTIKASI NT LAN MANAGER (NTLM)

Yoseph Suryadharma – NIM. 13504037

Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung
Email: if14037@students.if.itb.ac.id

Abstrak

Proses otentikasi berperan sangat penting untuk komunikasi di dalam suatu jaringan. Proses otentikasi jaringan menggunakan suatu protokol kriptografi. Proses ini berguna untuk mencegah agar tidak sembarang orang bisa melakukan komunikasi dalam jaringan. Cara yang sering digunakan dalam melakukan otentikasi adalah dengan menggunakan sandi-lewat (*password*). Banyak protokol yang digunakan dalam melakukan otentikasi dalam komunikasi jaringan. Salah satu protokol yang banyak digunakan adalah NT LAN Manager (NTLM).

NTLM adalah suatu protokol otentikasi yang digunakan dalam implementasi protokol jaringan Microsoft dan didukung oleh NTLM Security Support Provider (NTLMSSP). NTLM menggantikan protokol otentikasi jaringan LAN Manager karena protokol LAN Manager sudah dianggap tidak aman lagi. Protokol NTLM menggunakan *challenge-response sequence* yang memerlukan tiga kali pengiriman pesan antara *client* dan *server*. Tiga pesan ini dikenal sebagai Tipe I (*negotiate*), Tipe II (*challenge*), dan Tipe III (*authentication*). Protokol otentikasi NTLM menggunakan algoritma hash MD4/MD5 dan algoritma enkripsi DES.

Walaupun protokol otentikasi NTLM lebih baik dari protokol sebelumnya, yaitu LAN Manager, protokol ini juga tidak luput dari berbagai serangan. Serangan yang sering digunakan untuk menembus protokol NTLM adalah *dictionary attacks* dan *middle person attacks*. Untuk menanggulangi hal ini, protokol NTLM terus dikembangkan menjadi beberapa versi.

Kata kunci: protokol, otentikasi, NT LAN Manager (NTLM), *challenge-response sequence*.

1. Pendahuluan

Protokol adalah aturan yang berisi rangkaian langkah-langkah, yang melibatkan dua atau lebih orang, yang dibuat untuk menyelesaikan suatu kegiatan. Sedangkan protokol kriptografi adalah protokol yang menggunakan kriptografi . Orang yang berpartisipasi dalam protokol kriptografi memerlukan protokol tersebut misalnya untuk berbagi komponen rahasia untuk menghitung sebuah nilai, membangkitkan rangkaian bilangan acak, meyakinkan identitas orang lain atau otentikasi, dan sebagainya.

Protokol kriptografi dibangun melibatkan beberapa algoritma kriptografi. Sebagian besar protokol kriptografi dirancang untuk dipakai oleh kelompok yang terdiri dari dua orang pemakai, tetapi ada juga beberapa protokol yang dirancang untuk dipakai oleh kelompok yang terdiri lebih dari dua orang pemakai. Di dalam praktiknya, pihak yang terlibat dalam suatu protokol tidak harus orang, tetapi bisa juga mesin. Misalnya antara komputer *client* dan komputer *server* dalam suatu jaringan.

Dalam berkomunikasi di dalam suatu jaringan, pihak-pihak yang terlibat harus melalui suatu proses yang disebut sebagai proses otentikasi. Proses ini sangat penting dalam komunikasi agar tidak sembarang orang bisa melakukan komunikasi. Dengan proses otentikasi, identitas pihak yang berkomunikasi bisa dipastikan. Proses otentikasi ini dirancang sedemikian rupa sehingga cukup aman dan sukar ditembus oleh pihak-pihak yang tidak berwenang untuk melakukan komunikasi.

Terdapat bermacam-macam protokol otentikasi yang digunakan dalam komunikasi di suatu jaringan, salah satunya adalah NT LAN Manager (NTLM). NT LAN Manager (NTLM) adalah protokol otentikasi yang sering digunakan dalam jaringan ber-*platform* Windows. Protokol jaringan yang diperkenalkan oleh Microsoft ini didukung oleh NT Lan Manager Security Support Provider (NTLMSSP). NTLM menggantikan protokol otentikasi jaringan LAN Manager karena LAN Manager sudah dianggap tidak aman lagi.

Protokol NTLM menjadi protokol *default* untuk otentikasi jaringan pada Windows NT 4.0. Protokol NTLM menggunakan *challenge-response sequence*

untuk otentikasi *client*. Proses ini memerlukan tiga kali pengiriman pesan antara *client* dan *server*. Tiga pesan ini dikenal sebagai Tipe I (*negotiation*), Tipe II (*challenge*), dan Tipe III (*authentication*). Protokol ini menggunakan algoritma hash MD4/MD5 dan algoritma enkripsi DES.

Mekanisme otentikasi NTLM digunakan dalam konfigurasi berikut:

1. Windows 2000 Professional *client* mengotentikasi Windows NT 4.0 *domain controller*.
2. Microsoft Windows NT Workstation 4.0 *client* mengotentikasi Windows 2000 *domain controller*.
3. Windows NT Workstation 4.0 *client* mengotentikasi Windows NT 4.0 *domain controller*.
4. *User* Windows NT 4.0 *domain* mengotentikasi Windows 2000 *domain*.

2. Algoritma Protokol NT LAN Manager (NTLM)

Protokol NTLM menggunakan mekanisme *challenge-response sequence* dalam melakukan otentikasi. Dengan mekanisme ini, memungkinkan *client* untuk membuktikan identitasnya tanpa perlu mengirimkan sandi-lewat (*password*) kepada *server*. Mekanisme ini terdiri atas tiga kali proses pengiriman pesan antara *client-server*. Pesan yang dikirim dikenal sebagai Tipe I (*negotiation*), Tipe II (*challenge*), dan Tipe III (*authentication*).

Secara garis besar, mekanisme yang digunakan oleh protokol otentikasi NTLM adalah sebagai berikut:

1. *Client* mengirimkan pesan Tipe I yang berisi sekumpulan *flag-flag* dari fitur-fitur yang didukung atau diminta (seperti ukuran kunci enkripsi, permintaan otentikasi timbal-balik, dll) kepada *server*.
2. *Server* memberikan respon dengan mengirimkan pesan Tipe II yang berisi sekumpulan *flag-flag* yang mirip, yang didukung atau diperlukan oleh *server* (melakukan persetujuan mengenai parameter otentikasi antara *client-server*), dan yang lebih penting yaitu *challenge* acak (8 *bytes*).
3. Kemudian *client* menggunakan *challenge* yang ada pada pesan Tipe II dan mandat pengguna untuk menghitung respon. Metode penghitungan ini berbeda-beda tergantung dari parameter otentikasi NTLM yang telah dinegosiasikan sebelumnya, secara umum menggunakan algoritma hash MD4/MD5 dan algoritma enkripsi DES untuk menghitung respon. *Client* kemudian mengirim respon ini menggunakan pesan Tipe III kepada *server*.

2.1 *Flag* pada NTLM

Flag-flag NTLM terletak dalam *bitfield* pada *header* pesan. *Flag* ini bertipe *long*, setiap bit merepresentasikan *flag* yang spesifik. Macam-macam *flag* yang terdapat pada NTLM terdapat pada tabel berikut:

Flag	Nama	Deskripsi
0x00000001	Negotiate Unicode	Menandakan bahwa *string* Unicode didukung untuk penggunaan pada *security buffer data*
0x00000002	Negotiate OEM	Menandakan bahwa *string* OEM didukung untuk penggunaan pada *security buffer data*
0x00000004	Request Target	Meminta agar otentikasi server dalam dimasukkan dalam pesan Tipe II
0x00000008	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x00000010	Negotiate Sign	Menspesifikasikan bahwa komunikasi yang telah diotentikasi antara *client-server* harus menggunakan tanda tangan digital
0x00000020	Negotiate Seal	Menspesifikasikan bahwa komunikasi yang telah diotentikasi antara *client-server* harus dienkripsi.
0x00000040	Negotiate Datagram Seal	Menandakan bahwa otentikasi datagram sedang digunakan
0x00000080	Negotiate LAN Manager Key	Menandakan bahwa LAN Manager *session key* harus digunakan untuk menandai dan menyegel

		komunikasi yang sudah diotentikasi
0x00000100	Negotiate Netware	Penggunaan *flag* ini belum diidentifikasi
0x00000200	Negotiate NTLM	Menandakan bahwa otentikasi
0x00000400	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x00000800	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x00001000	Negotiate Domain Supplied	Dikirim oleh *client* dalam bentuk pesan Tipe I untuk menandakan bahwa nama domain dari workstation dimasukkan ke dalam pesan. Digunakan oleh *server* untuk menentukan apakah *client* layak untuk otentikasi lokal
0x00002000	Negotiate Workstation Supplied	Dikirim oleh *client* dalam bentuk pesan Tipe I untuk menandakan bahwa nama workstation dimasukkan ke dalam pesan. Digunakan oleh *server* untuk menentukan apakah *client* layak untuk otentikasi lokal
0x00004000	Negotiate Local Call	Dikirim oleh *server* menandakan bahwa *server* dan *client* berada pada mesin yang sama
0x00008000	Negotiate Always Sign	Menandakan bahwa komunikasi yang diotentikasi antara *client* dan *server* ditandai dengan *dummy signature*
0x00010000	Target Tipe Domain	Dikirim oleh *server* dalam pesan Tipe II yang menandakan bahwa target otentikasi adalah sebuah domain
0x00020000	Target Tipe Server	Dikirim oleh *server* dalam pesan Tipe II yang menandakan bahwa target otentikasi adalah sebuah *server*
0x00040000	Target Tipe Share	Dikirim oleh *server* dalam pesan Tipe II yang menandakan bahwa target otentikasi adalah sebuah *share computer*
0x00080000	Negotiate NTLM2 Key	Menandakan bahwa skema penandatanganan dan penyegelan NTLM2 harus digunakan untuk melindungi komunikasi diotentikasi
0x00100000	Request Init Response	Penggunaan *flag* ini belum diidentifikasi
0x00200000	Request Accept Response	Penggunaan *flag* ini belum diidentifikasi
0x00400000	Request Non-NT Session Key	Penggunaan *flag* ini belum diidentifikasi
0x00800000	Negotiate Target Info	Dikirim oleh *server* dalam pesan Tipe II untuk menandakan bahwa pesan memiliki *Target Information block*
0x01000000	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x02000000	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x04000000	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x08000000	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x10000000	Tidak diketahui	Penggunaan *flag* ini belum diidentifikasi
0x20000000	Negotiate 128	Menandakan bahwa *client/server* mendukung enkripsi 128 bit
0x40000000	Negotiate Key Exchange	Menandakan bahwa *client* akan menyediakan *master session key* yang sudah dienkripsi di dalam pesan Tipe III
0x80000000	Negotiate 56	Menandakan bahwa *client/server* mendukung enkripsi 56 bit

2.2 Pesan Tipe I (*Negotiation*)

Pesan Tipe I berisi sebagai berikut:

	Deskripsi	Isi
0	NTLMSSP *signature*	*Null-terminated* ASCII "NTLMSSP" (0x4e544c4d53535000)
8	NTLM *Message Type*	*long* (0x01000000)
12	*Flags*	*long*
(16)	*Supplied Domain (optional)*	*Security buffer*
(24)	*Supplied Workstation (optional)*	*Security buffer*
(32)	*Start of data block* (jika diperlukan)	

Pesan Tipe I dikirim oleh *client* ke *server* untuk melakukan inisiasi otentikasi NTLM. Tujuan utamanya adalah untuk membangun aturan dasar untuk otentikasi dengan memberikan opsi yang didukung melalui *flag*. Pesan ini bisa juga untuk memberitahu *server* nama *workstation client* atau nama domain *client*. Informasi ini akan digunakan oleh *server* untuk menentukan apakah *client* layak untuk melakukan otentikasi lokal.

Secara khusus, pesan Tipe I akan mengandung *flag-flag* berikut:

1. Negotiate Unicode (0x00000001)
2. Negotiate OEM (0x00000002)
3. Request Target (0x00000004)
4. Negotiate NTLM (0x00000200)
5. Negotiate Domain Supplied (0x00001000)
6. Negotiate Workstation Supplied (0x00002000)
7. Negotiate Always Sign (0x00008000)
8. Negotiate NTLM2 Key (0x00080000)
9. Negotiate 128 (0x20000000)
10. Negotiate 56 (0x80000000)

Pesan Tipe I yang paling minimal adalah

```
4e544c4d53535000010000002020000
```

Pesan tersebut hanya berisi NTLMSSP *signature*, NTLM *message Tipe*, dan *flag* minimal (Negotiate NTLM dan Negotiate OEM).

Contoh pesan Tipe I adalah sebagai berikut

```
4e544c4d535350000100000073200000
06000600 2b0000000b000b0020000000 574f
524b53544154494f4e444f4d41494e
```

Pesan tersebut dipecah sebagai berikut

0	`0x4e544c4d53535000`	NTLMSSP *signature*
8	`0x01000000`	Indikator Tipe I
12	`0x07320000`	*Flags:* Negotiate Unicode (`0x00000001`) Negotiate OEM (`0x00000002`) Request Target (`0x00000004`) Negotiate NTLM (`0x00000200`) Negotiate Domain Supplied (`0x00001000`) Negotiate Workstation Supplied (`0x00002000`)
16	`0x060006002b000000`	*Domain Security Buffer*: Panjang: 6 *bytes* (`0x0600`) Alokasi: 6 *bytes* (`0x0600`) Offset: 43 *bytes* (`0x2b000000`)
24	`0x0b000b0020000000`	*Workstation Security Buffer:* Panjang: 11 *bytes* (`0x0b00`) Alokasi: 11 *bytes* (`0x0b00`) Offset: 32 *bytes* (`0x20000000`)
32	`0x574f524b53544154494f4e`	DataWorkstation ("WORKSTATION")
43	`0x444f4d41494e`	Data Domain ("DOMAIN")

Arti dari informasi tersebut adalah:

1. Pesan tersebut adalah pesan Tipe I (dari NTLMSSP *signature* dan indikator Tipe I).

2. *Client* mendukung *string* Unicode maupun OEM (dari *flag* Negotiate Unicode dan *flag* Negotiate OEM).
3. *Client* mendukung otentikasi NTLM (dari *flag* Negotiate NTLM).
4. *Client* meminta agar *server* mengirimkan informasi tentang target otentikasi (dari *flag* Request Target).
5. *Client* mengirimkan informasi tentang nama domainnya, yaitu "DOMAIN" (dari *flag* Negotiate Domain Supplied, informasi nama domain ada pada *Domain Security Buffer*).
6. *Client* mengirimkan informasi tentang nama *workstation*, yaitu "WORKSTATION" (dari *flag* Negotiate Workstation Supplied, informasi nama workstation berada pada *workstation Security Buffer*).

Setelah menciptakan pesan Tipe I, *client* mengirimkannya kepada *server*. *Server* kemudian akan memberikan respon dengan mengirimkan pesan Tipe II kepada *client*.

2.3 Pesan Tipe II (*Challenge*)

Pesan Tipe II berisi sebagai berikut:

	Deskripsi	Isi
0	NTLMSSP *signature*	*Null-terminated* ASCII "NTLMSSP" (0x4e544c4d53535000)
8	NTLM *message Type*	*long* (0x02000000)
12	*Target name*	*Security buffer*
20	*Flags*	*long*
24	*Challenge*	8 *bytes*
(32)	*Context (optional)*	8 *bytes*
(40)	*Target information*	*Security buffer*
32 (48)	*Start of data block*	

Pesan Tipe II dikirim oleh *server* sebagai respon terhadap pesan Tipe I yang dikirim oleh *client*. Pesan ini melengkapi proses negosiasi opsi-opsi dengan *client*, dan juga memberikan *challenge* kepada *client*. Secara opsional, pesan ini bisa mengandung informasi mengenai target otentikasi.

Pesan Tipe II akan memiliki *flag-flag* sebagai berikut:

1. Negotiate Unicode (0x00000001)
2. Negotiate OEM (0x00000002)
3. Request Target (0x00000004)
4. Negotiate NTLM (0x00000200)
5. Negotiate Local Call (0x00004000)
6. Negotiate Always Sign (0x00008000)
7. Target Tipe Domain (0x00010000)
8. Target Tipe Server (0x00020000)
9. Target Tipe Share (0x00040000)
10. Negotiate NTLM2 Key (0x00080000)
11. Negotiate Target Info (0x00800000)
12. Negotiate 128 (0x20000000)
13. Negotiate 56 (0x80000000)

Target name adalah sebuah *security buffer* yang menyimpan nama dari target otentikasi. *Target name* secara khusus dikirim sebagai respon terhadap permintaan target dari *client* (melalui *flag* Request Target pada pesan Tipe I). Info tersebut dapat berupa *domain*, *server*, atau sebuah *network share*. Tipe target dapat diketahui melalui *flag* Target Tipe Domain, Target Tipe Server, dan Target Tipe Share. *Target name* dapat berupa karakter Unicode atau OEM.

Challenge berupa 8 *byte* blok *random data* . *Challenge* akan digunakan oleh *client* untuk melakukan formulasi respon. Respon ini akan dikirim melalui pesan Tipe III.

Context field diciptakan ketika *flag* Negotiate Local Call diset. Berisi SSPI *context handler*, yang memungkinkan *client* untuk melakukan otentikasi "*short-circuit*" sehingga *client* bisa "mengelak" untuk merespon *challenge*.

Target information berupa *security buffer* yang mengandung blok *Target Information*. Berisi sekuens dari beberapa subblok yang masing-masing berisi:

Field	Isi	Deskripsi
Tipe	*short*	Menandakan tipe data: (0x0100) nama server (0x0200) nama domain (0x0300) nama *host* DNS (0x0400) nama domain DNS
Panjang	*short*	Panjang dari subblok
Isi	*string* Unicode	Isi dari subblok

Sekuens ini diterminasi oleh subblok terminasi, bertipe "0" dengan panjang nol.

Pesan Tipe II yang paling minimal adalah:

```
4e544c4d535350000200000000000000
0000000202000001234567 89abcdef
```

Pesan tersebut hanya mengandung NTLMSSP *signature*, NTLM *message Tipe*, *target name* kosong, *flag* minimal (Negotiate NTLM dan Negotiate OEM), dan *challenge*.

Contoh pesan Tipe II adalah sebagai berikut:

```
4e544c4d53535000020000000c000c0030
00000001028100012345678 9abcdef0000
0000000000000620062003c00000044004f
004d00410049004e0002000c0044004f00
4d00410049004e0001000c005300450052
00560045005200040014006 4006f006d00
610069006e002e0063006f006d00030022
00730065007200760065 0072002e006400
6f006d00610069006e002e0063006f006d
0000000000
```

Pesan tersebut dipecah menjadi:

0	`0x4e544c4d 53535000`	NTLMSSP *signature*
8	`0x02000000`	Indikator Tipe II
12	`0x0c000c00 30000000`	*Target name security buffer:* Panjang: 12 *bytes* (`0x0c00`) Alokasi: 12 *bytes* (`0x0c00`) *Offset*: 48 *bytes* (`0x30000000`)
20	`0x01028100`	*Flags*: Negotiate Unicode (`0x00000001`) Negotiate NTLM (`0x00000200`) Target Tipe Domain (`0x00010000`) Negotiate Target Info (`0x00800000`)
24	`0x01234567 89abcdef`	*Challenge*
32	`0x00000000 00000000`	*Context*
40	`0x62006200 3c000000`	*Target information security buffer:* Panjang: 98 *bytes* (`0x6200`) Alokasi: 98 *bytes* (`0x6200`) *Offset*: 60 *bytes* (`0x3c000000`)
48	`0x44004f00 4d00410049 004e00`	*Target Name Data* (`"DOMAIN"`)
60	`0x02000c00 44004f004d`	*Target Information Data:*

	`0041004900 4e0001000c 0053004500 5200560045 0052000400 140064006f 006d006100 69006e002e 0063006f00 6d00030022 0073006500 7200760065 0072002e00 64006f006d 0061006900 6e002e0063 006f006d00 00000000`	`0x0200 0c0044 004f00 4d0041 004900 4e00`	*Domain name subblock* Tipe: 2 (Domain name, 0x0200) Panjang: 12 *bytes* (0x0c00) Data: "DOMAIN"
		`0x0100 0c0053 004500 520056 004500 5200`	*Server name subblock*: Tipe: 1 (Server name, 0x0100) Panjang: 12 *bytes* (0x0c00) Data: "SERVER"
		`0x0400 140064 006f00 6d0061 006900 6e002e 006300 6f006d 00`	*DNS domain name subblock:*Tipe: 4 (DNS domain name, `0x0400`) Panjang: 20 *bytes* (`0x1400`) Data: `"domain.com"`
		`0x0300 220073 006500 720076 006500 72002e 006400 6f006d 006100 69006e 002e00 63006f 006d00`	*DNS server name subblock:* Tipe: 3 (DNS server name, `0x0300`) Panjang: 34 bytes (`0x2200`) Data: `"server.domain.com"`
		`0x0000 0000`	*Terminator subblock:*

			Tipe: 0 (terminator, 0x0000) Panjang: 0 bytes (0x0000)

Arti dari pesan tersebut adalah sebagai berikut:

1. Pesan tersebut adalah pesan Tipe II (dari NTLMSSP *signature* dan indikator tipe II).
2. *Server* memberitahu bahwa *string* akan dikodekan dalam Unicode (dari *flag* Negotiate Unicode).
3. *Server* mendukung otentikasi NTLM (Negotiate NTLM).
4. *Target name* disediakan oleh *server* dan merepresentasikan suatu *domain* (dari *flag* Target Tipe Domain, dan nama domain berada pada *target name security buffer*).
5. *Server* menyediakan struktur *Target information* (dari *flag* Negotiate Target Info). Struktur ini terdapat pada *target information security buffer*.
6. *Challenge* yang dibangkitkan oleh *server* adalah "0x0123456789abcdef".
7. *Empty context* telah dikirim.

Setelah *server* membuat pesan Tipe II, pesan tersebut dikirimkan kepada *client*. Respon *challenge* dikirim oleh *client* dalam pesan Tipe III.

2.4 Pesan Tipe III (*Authentication*)

Pesan Tipe III berisi sebagai berikut:

	Deskripsi	Isi
0	NTLMSSP *signature*	Null-terminated ASCII "NTLMSSP" (0x4e544c4d53535000)
8	NTLM *Message Type*	long (0x03000000)
12	LM/LMv2 *Response*	security buffer
20	NTLM/NTLM v2 *Response*	security buffer
28	*Domain Name*	security buffer
36	*User Name*	security buffer
44	*Workstation Name*	security buffer
(52)	*Session Key (optional)*	security buffer
(60)	*Flags (optional)*	long
52(64)	*start of data block*	

Pesan Tipe III adalah langkah akhir proses otentikasi. Pesan ini berisi respon *client* terhadap *challenge* pada pesan Tipe II , yang menunjukkan bahwa *client* memiliki informasi mengenai sandi-lewat (*password*) tanpa mengirimkannya secara langsung kepada *server*. Pesan Tipe III juga menginformasikan mengenai nama *domain*, *username*, dan nama *workstation* milik *client*.

Flag pada pesan Tipe III bersifat opsional; *client* lama tidak memasukkan baik *session key* maupun . Pada kasus ini, blok data berawal pada *offset* 52. Secara eksperimental telah ditentukan bahwa *flag* Tipe III, tidak membawa suatu semantik tambahan di dalam otentikasi berorientasi koneksi. *Client* mengirimkan *flag* sebagai *mirror* dari *flag* pada pesan Tipe II. *Flag* dikirim sebagai pengingat mengenai opsi-opsi yang telah disetujui bersama supaya *server* tidak perlu melakukan *caching* opsi-opsi yang telah dinegosiasikan.

LM/LMv2 dan NTLM/NTLMv2 adalah sebuah *security buffer* yang berisi jawaban yang dibuat dari sandi-lewat (*password*) *user* sebagai respon terhadap *challenge*.

Nilai dari *session key* tidak diketahui, dan mungkin kosong. Hal ini relevan dengan mekanisme penandaan dan penyegelan yang baru.

Ketika *flag* Negotiate Local Call diset pada pesan Tipe II, semua *security buffer* pada pesan Tipe III kosong. *Client* mengadopsi SSPI *context* yang ada pada pesan Tipe II, memungkinkan *client* untuk mengelak keharusan memberikan respon yang tepat.

2.4.1 Respon Terhadap *Challenge*

Client menciptakan satu atau lebih respon terhadap *challenge*, dan mengirimkannya dalam pesan Tipe III. Terdapat lima jenis respon sebagai berikut:

1. LM (LAN Manager) *response*
2. NTLM *response*
3. NTLMv2 *response*
4. LMv2 *response*
5. NTLM2 *session response*

Gambaran respon tersebut secara detil adalah sebagai berikut:

(1) LM (LAN Manager) *response*

Respon ini dikirim oleh kebanyakan *client*. Skema ini lebih tua dari NTLM dan lebih tidak aman. Jika *client* sudah mendukung respon NTLM, mereka akan mengirimkan kedua jenis respon ini untuk menjaga kompatibilitas dengan *server*.

Penghitungan respon LM adalah sebagai berikut:

1. Sandi-lewat (*password*) *user* dikonversikan ke dalam huruf kapital.
2. Sandi-lewat (*password*) tersebut ditambah atau dipangkas menjadi 14 *bytes*.
3. Kemudian sandi-lewat (*password*) tersebut dibagi menjadi dua bagian masing-masing berukuran 7 *bytes.*
4. Nilai tersebut lalu digunakan untuk membangitkan dua buah kunci DES.
5. Setiap kunci tersebut digunakan untuk mengenkripsi dengan algoritma DES konstanta ASCII `"KGS!@#$%"` (menghasilkan dua buah *ciphertext* berukuran 8 *byte*).
6. Kedua *ciphertext* tersebut dikonkatenasi untuk membentuk sebuah nilai LM hash 16 *bytes*.
7. Kemudian nilai LM hash tersebut ditambah menjadi 21 *bytes*.
8. Nilai tersebut kemudian dibagi menjadi tiga bagian masing-masing berukuran 7 *bytes*.
9. Nilai-nilai tersebut kemudian digunakan untuk membangkitkan tiga buah kunci DES.
10. Setiap kunci tersebut digunakan untuk mengenkripsi *challenge* pada pesan Tipe II (menghasilkan tiga buah *ciphertext* berukuran 8 *bytes*).
11. Kemudian hasil tersebut dikonkatenasi menjadi nilai berukuran 24 *bytes.*

Sebagai contoh, misalkan sandi-lewat pengguna adalah `"SecREt01"`, akan melakukan respon terhadap *challenge* `"0x0123456789abcdef"`. Langkah-langkah penghitungan responnya adalah sebagai berikut:

1. Sandi-lewat dikonversi ke huruf kapital menjadi `"SECRET01"` (atau `"0x5345435245543031"` dalam heksa desimal).
2. Kemudian ditambah dengan *padding bit* menjadi berukuran 14 *bytes* menjadi `"0x534543524554303100000000000000"`.
3. Setelah itu dibagi menjadi dua masing-masing berukuran 7 *bytes*, yaitu `"0x53454352455430"` dan `"0x31000000000000".`
4. Kedua nilai tersebut kemudian dipakai untuk menentukan dua buah kunci DES. Nilai pertama `0x53454352455430"` dalam biner menjadi:

   ```
   01010011  01000101  01000011
   01010010  01000101  01010100
   00110000
   ```

   Kunci DES *non-parity* untuk nilai tersebut adalah:

   ```
   01010010  10100010  01010000
   01101010  00100100  00101010
   01010000 01100000
   ```

   Dalam heksadesimal adalah `"0x52a2506a242a5060"`. Dengan menerapkan *parity* ganjil untuk meyakinkan agar setiap sekuens *bit* bernilai ganjil menghasilkan:

   ```
   01010010  10100010  01010001
   01101011  00100101  00101010
   01010001 01100001
   ```

   Nilai tersebut adalah kunci pertama DES, dalam heksadesimal bernilai `"0x52a2516b252a5161"`. Kemudian lakukan proses tadi untuk nilai `"0x31000000000000"`, dalam biner bernilai:

   ```
   00110001  00000000  00000000
   00000000  00000000  00000000
   00000000
   ```

Kunci DES *non-parity* untuk nilai tersebut adalah:

```
00110000  10000000  00000000
00000000  00000000  00000000
00000000 00000000
```

Dalam heksadesimal bernilai `"0x3080000000000000"`. Dengan mengatur *parity* agar ganjil, menghasilkan:

```
00110001  10000000  00000001
00000001  00000001  00000001
00000001 00000001
```

Nilai tersebut adalah kunci DES kedua, dalam heksadesimal bernilai `"0x3180010101010101"`.

5. Kemudian setiap kunci tersebut digunakan untuk mengenkripsi konstanta ASCII `"KGS!@#$%"` (`"0x4b47532140232425"` dalam heksadesimal). Menghasilkan `"0xff3750bcc2b22412"` (menggunakan kunci pertama) dan `"0xc2265b23734e0dac"` (menggunakan kunci kedua).
6. Kemudian dikonkatenasi untuk membangun 16 *bytes* LM hash menjadi `"0xff3750bcc2b22412c2265b23734e0dac"`.
7. Kemudian ditambahi menjadi 21 *bytes*, menjadi `"0xff3750bcc2b22412c2265b23734e0dac0000000000"`.
8. Kemudian dibagi menjadi tiga bagian yaitu `"0xff3750bcc2b224"`, `"0x12c2265b23734e"` dan `"0x0dac0000000000"`.
9. Ketiga nilai tersebut digunakan untuk membangkitkan tiga buah kunci DES seperti pada proses sebelumnya menghasilkan kunci-kunci `"0xfe9bd516cd15c849"`, `"0x136189cbb31acd9d"`, `"0x0dd6010101010101"`.
10. Kemudian kunci-kunci tersebut digunakan untuk mengenkripsi nilai *challenge* pada pesan Tipe II `"0x0123456789abcdef"`. Hasil enkripsinya adalah `"0xc337cd5cbd44fc97"` (menggunakan kunci pertama), `"0x82a667af6d427c6d"` (menggunakan kunci kedua) dan `"0xe67c20c2d3e77c56"` (menggunakan kunci ketiga).
11. Kemudian nilai tersebut dikonkatenasi menghasilkan 24 *bytes* nilai respon LM, yaitu:

```
0xc337cd5cbd44fc9782a667af6d
427c6d e67c20c2d3e77c56
```

(2) NTLM *response*

Penghitungan respon NTLM adalah sebagai berikut:

1. Algoritma MD4 diaplikasikan ke dalam sandi-lewat *user* menghasilkan 16 *bytes* nilai NTLM hash.
2. NTLM hash tersebut ditambah dengan *padding bit* menjadi 21 *bytes*.
3. Kemudian nilai tersebut dibagi menjadi tiga bagian masing-masing berukuran 7 *bytes*.
4. Nilai-nilai tersebut digunakan untuk menghitung tiga buah kunci DES.
5. Setiap kunci tersebut kemudian digunakan untuk mengenkripsi *challenge* (menghasilkan tiga buah *ciphertext* berukuran 8 *bytes*).
6. Kemudian hasil enkripsi tersebut dikonkatenasi menjadi 24 *bytes*.

Sebagai contoh, misalkan sandi-lewat *user* adalah `"SecREt01"`, akan memberikan respon terhadap *challenge* `"0x0123456789abcdef"`. Langkah-langkah penghitungan responnya adalah sebagai berikut:

1. Dalam heksadesimal nilai sandi-lewat *user* adalah `"0x5300650063005200450074003 0003100"`, kemudian dihitung nilai hashnya menggunakan MD4 menghasilkan `"0xcd06ca7c7e10c99b1d33b7485a2ed808"`.
2. Nilai tersebut ditambah dengan *padding bit* menjadi 21 *bytes* menjadi `"0xcd06ca7c7e10c99b1d33b7485a2ed8080000000000"`.

3. Nilai tersebut dibagi menjadi tiga bagian berukuran 7 *bytes*, menghasilkan `"0xcd06ca7c7e10c9"`, `"0x9b1d33b7485a2e"` dan `"0xd8080000000000"`.
4. Ketiga nilai tersebut digunakan untuk menghitung tiga buah kunci DES. Nilai pertama dalam bit adalah:

   ```
   11001101  00000110  11001010
   01111100  01111110  00010000
   11001001
   ```

   Menggunakan penyesuaian *parity* akan menghasilkan kunci

   ```
   11001101  10000011  10110011
   01001111  11000111  11110001
   01000011 10010010
   ```

   Dalam heksadesimal bernilai `"0xcd83b34fc7f14392"`. Nilai kedua dalam *bit* bernilai

   ```
   10011011  00011101  00110011
   10110111  01001000  01011010
   00101110
   ```

   Akan menghasilkan kunci

   ```
   10011011  10001111  01001100
   01110110  01110101  01000011
   01101000 01011101
   ```

   Dalam heksadesimal bernilai `"0x9b8f4c767543685d"`. Nilai ketiga

   ```
   11011000  00001000  00000000
   00000000  00000000  00000000
   00000000
   ```

   Menghasilkan kunci

   ```
   11011001  00000100  00000001
   00000001  00000001  00000001
   00000001 00000001
   ```

   Dalam heksadesimal bernilai `"0xd9040101010101"`.
5. Ketiga kunci tersebut kemudian digunakan untuk mengenkripsi nilai *challenge* `"0x0123456789abcdef"` menghasilkan `"0x25a98c1c31e81847"` (menggunakan kunci pertama), `"0x466b29b2df4680f3"` (menggunakan kunci kedua) dan `"0x9958fb8c213a9cc6"` (menggunakan kunci ketiga).
6. Nilai-nilai tersebut kemudian dikonkatenasi menjadi 24 *bytes* (nilai inilah yang disebut NTLM *response*) menjadi **`0x25a98c1c31e81847466b29b2df4680f39958fb8c213a9cc6`**

(3) NTLMv2 *response*

NTLMv2 dibuat untuk memperbarui NTLM agar lebih aman. Ketika *client* menggunakan NTLMv2 maka respon LM akan digantikan dengan LMv2. Algoritma penghitungan respon NTLMv2 adalah sebagai berikut:

1. Nilai hash dari sandi-lewat *user* dicari menggunakan MD4 seperti pada respon NTLM.
2. Karakter Unicode *username* (semua dalam huruf kapital) dikonkatenasi dengan karakter Unicode target otentikasi (nama *domain* atau *server* dalam huruf kapital). Kemudian dicari nilai hashnya menggunakan HMAC-MD5dengan menggunakan nilai NTLM hash yang telah didapat sebagai kunci menghasilkan 16 *bytes* nilai NTLMv2 hash.
3. Mengkonstruksi blok data yang dikenal sebagai ‘blob’, yang berisi sebagai berikut:

	Deskripsi	Isi
0	Blob *signature*	`0x01010000`
4	*Reserved*	long (`0x00000000`)
8	*Timestamp*	64 *bit signed value* yang merepresentasikan jumlah sepersepuluh dari mikrodetik sejak 1 Januari 1601
16	*Client challenge*	8 *bytes*
24	Tidak diketahui	4 *bytes*

28	*Target Information*	Blok informasi target
variable	Tidak diketahui	4 *bytes*

4. *Challenge* kemudian dikonkatenasi dengan blob. Kemudian menghitung nilai hash menggunakan HMAC-MD5 dengan nilai NTLMv2 hash sebagai kuncinya menghasilkan keluaran 16 *bytes*.
5. Nilai tersebut kemudian dikonkatenasi dengan blob.Nilai inilah yang disebut respon NTLMv2.

Sebagai contoh, misalkan digunakan informasi sebagai berikut:

Domain	`DOMAIN`
Username	`user`
Password	`SecREt01`
Challenge	`0x0123456789abcdef`
Target Information	`0x02000c0044004f004d00410049004e0001000c00530045005200560045005200400400140064006f006d00610069006e002e0063006f006d0003002200730065007200760065007200 2e0064006f006d00610069006e002e0063006f006d0000000000`

1. Nilai *password* dalam heksadesimal adalah `"0x5300650063005200450074003 0003100"`. Nilai MD4nya adalah `"0xcd06ca7c7e10c99b1d33b7485 a2ed808"`. Nilai tersebut adalah nilai NTLM hash.
2. Karakter Unicode *username* dikonkatenasi dengan nama domain dan dikonversi ke dalam huruf kapital menjadi `"USERDOMAIN"`; dalam heksadesimal bernilai `"0x550053004500520044004f004 d00410049004e00"`. Kemudian menerapkan algoritma HMAC-MD5 dengan nilai NTLM hash sebagai kunci menghasilkan `"0x04b8e0ba74289cc540826bab1 dee63ae"`. Nilai tersebut adalah NTLMv2 hash.
3. Kemudian melakukan konstruksi blob dengan nilai misalkan adalah sebagai berikut:

`0x01010000`
`0x00000000`
`0x0090d336b734c301`
`0xffffff0011223344`
`0x00000000`
`0x02000c0044004f004d004 10049004e0001000c005300 4500520056004500520040 0140064006f006d00610069 006e002e0063006f006d000 3002200730065007200760 0650072002e0064006f006d0 0610069006e002e0063006f 006d0000000000`
`0x00000000`

4. Kemudian melakukan konkatenasi nilai *challenge* dengan blob menghasilkan:

   `0x0123456789abcdef0101000000 0000000090d336b734c301ffffff 0011223344000000002000c0044 004f004d00410049004e0001000c 0053004500520056004500520004 00140064006f006d00610069006e 002e0063006f006d000300220073 0065007200760065007200 2e0064006f006d00610069006e002e0063 006f006d000000000000000000`

   Kemudian menerapkan algoritma HMAC-MD5 dengan nilai NTLMv2 hash sebagai kunci menghasilkan nilai `"0xcbabbca713eb795d04c97abc0 1ee4983"`.

5. Kemudian nilai tersebut dikonkatenasi dengan blob untuk mendapatkan nilai respon NTLMv2 yaitu:

   `0xcbabbca713eb795d04c97abc0 1ee498301010000000000000090 d336b734c301 ffffff0011223344000000002020 00c0044004f004d00410049004e 0001000c0053004500520056004 5005200040014006400 6f006d00 610069006e002e0063006f006d0 0300220073006500720076006 5 0072002e0064006f006d0061006 9006e002e0063006f006d000000 000000000000`

(4) LMv2 *response*

LMv2 diciptakan untuk menjaga kompatibilitas dengan *server* lama. LMv2 merupakan suatu 'miniatur' NTLMv2. Algoritma penghitungan respon LMv2 adalah sebagai berikut:

1. Menghitung nilai NTLM hash dari sandi-lewat *user* menggunakan MD4.
2. Karakter Unicode *username* (semua dalam huruf kapital) dikonkatenasi dengan karakter Unicode target otentikasi (nama *domain* atau *server* dalam huruf kapital). Kemudian dicari nilai hashnya menggunakan HMAC-MD5dengan menggunakan nilai NTLM hash yang telah didapat sebagai kunci menghasilkan 16 *bytes* nilai NTLMv2 hash.
3. Membangkitkan *client challenge* berukuran 8 *bytes* secara acak.
4. *Challenge* pada pesan Tipe II dikonkatenasi dengan *challenge* yang telah dibangkitkan. Kemudian menerapkan algoritma HMAC-MD5 menggunakan NTLMv2 hash sebagai kunci, menghasilkan keluaran 16 *bytes*.
5. Nilai tersebut dikonkatenasi dengan nilai *client challenge* untuk membentuk respon LMv2 berukuran 24 *bytes*.

Sebagai ilustrasi, digunakan informasi sebagai berikut:

Domain	`DOMAIN`
Username	`user`
Password	`SecREt01`
Challenge	`0x0123456789abcdef`

1. Nilai *password* dalam heksadesimal adalah `"0x5300650063005200450074003 0003100"`. Nilai MD4nya adalah `"0xcd06ca7c7e10c99b1d33b7485 a2ed808"`. Nilai tersebut adalah nilai NTLM hash.
2. Karakter Unicode *username* dikonkatenasi dengan nama domain dan dikonversi ke dalam huruf kapital menjadi `"USERDOMAIN"`; dalam heksadesimal bernilai `"0x550053004500520044004f004 d00410049004e00"`. Kemudian menerapkan algoritma HMAC-MD5 dengan nilai NTLM hash sebagai kunci menghasilkan `"0x04b8e0ba74289cc540826bab1 dee63ae"`. Nilai tersebut adalah NTLMv2 hash.
3. Kemudian membangkitkan nilai *client challenge* berukuran 8 *bytes* secara acak, misalkan `"0xffffff0011223344"`.
4. Kemudian melakukan konkatenasi antara *challenge* pada pesan Tipe II dengan *client challenge*, menghasilkan `0x0123456789abcdefffffff0011 223344`. Kemudian menggunakan algoritma HMAC-MD5 dengan nilai NTLMv2 sebagai kunci akan menghasilkan nilai `"0xd6e6152ea25d03b7c6ba6629c 2d6aaf0"`.
5. Nilai tersebut kemudian dikonkatenasi dengan nilai *client challenge* untuk menghasilkan nilai respon LMv2 berukuran 24 *bytes* sebagai berikut

   `0xd6e6152ea25d03b7c6ba6629c2 d6aaf0ffffff0011223344`.

(5) NTLM2 *session response*

NTLM2 *session response* bisa digunakan dengan kombinasi dengan NTLM2 *session security* (dengan mengatur *flag* Negotiate NTLM2 Key). Digunakan untuk melindungi dari *dictionary attacks* pada lingkungan yang tidak mendukung otentikasi NTLMv2.

Penghitungan responnya adalah sebagai berikut:

1. Membangkitkan 8 *byte client challenge* secara acak.
2. Kemudian nilai tersebut ditambah dengan *padding bit* menjadi 24 *bytes*. Nilai ini ditempatkan dalam *field* respon LM pada pesan Tipe III.
3. *Challenge* pada pesan Tipe II dikonkatenasi dengan 8 *bytes* nilai *client challenge* membentuk *session nonce*.
4. Kemudian menerapkan algoritma MD5pada *session nonce* untuk menghasilkan keluaran 16 *bytes*.
5. Nilai tersebut kemudian dipangkas menjadi 8 *bytes* membentuk NTLM2 *session* hash.
6. Menghitung NTLM hash seperti pada metode respon NTLM.
7. Nilai 16 *bytes* NTLM hash ditambah dengan *padding bit* menjadi 21 *bytes*.
8. Nilai tersebut dipecah menjadi tiga bagian masing-masing berukuran 7 *bytes*.
9. Nilai-nilai tersebut kemudian digunakan untuk menghitung tiga buah nilai kunci DES.
10. Setiap kunci kemudian digunakan untuk mengenkripsi nilai NTLM2 *session hash* menghasilkan tiga buah nilai berukuran 8 *bytes*.

11. Ketiga nilai tersebut kemudian dikonkatenasi menjadi 24 *bytes*. Nilai tersebut adalah NTLM2 *session response* dan ditempatkan pada *field* respon NTLM pada pesan Tipe III.

2.4.2 Contoh Pesan Tipe III

Sebagai contoh, misalkan pesan Tipe III berisi sebagai berikut:

```
4e544c4d535350000300000018001800 6a
0000001800180082000000c000c004000
0000080008004c0000001600160054000 0
00000000009a0000000102000044004f00
4d00410049004e00750073006500720057
004f0052004b00530054004100540049 00
4f004e00c337cd5cbd44fc9782a667af6d
427c6de67c20c2d3e77c5625a98c1c31e8
1847466b29b2df4680f39958fb8c213a9c
c6
```

Pesan tersebut didekomposisi menjadi:

0	`0x4e544c4d53535000`	NTLMSSP *signature*
8	`0x03000000`	Indikator Tipe III
12	`0x180018006a000000`	LM *Response Security Buffer:* Panjang: 24 *bytes* (`0x1800`) Alokasi: 24 *bytes* (`0x1800`) *Offset*: 106 *bytes* (`0x6a000000`)
20	`0x1800180082000000`	NTLM *Response Security Buffer:* Panjang: 24 *bytes* (`0x1800`) Alokasi: 24 *bytes* (`0x1800`) *Offset*: 130 *bytes* (`0x82000000`)
28	`0x0c000c0040000000`	*Domain Name Security Buffer:* Panjang: 12 *bytes* (`0x0c00`) Alokasi: 12 *bytes* (`0x0c00`) *Offset*: 64 *bytes* (`0x40000000`)
36	`0x080008004c000000`	*User Name Security Buffer:* Panjang: 8 *bytes* (`0x0800`) Alokasi: 8 *bytes* (`0x0800`) *Offset*: 76 *bytes* (`0x4c000000`)
44	`0x1600160054000000`	*Workstation Name Security Buffer:* Panjang: 22 *bytes* (`0x1600`) Alokasi: 22 *bytes* (`0x1600`) *Offset*: 84 *bytes* (`0x54000000`)
52	`0x000000009a000000`	*Session Key Security Buffer:* Panjang: 0 *bytes* (`0x0000`) Alokasi: 0 *bytes* (`0x0000`) *Offset*: 154 *bytes* (`0x9a000000`)
60	`0x01020000`	*Flags:* Negotiate Unicode (`0x00000001`) Negotiate NTLM (`0x00000200`)
64	`0x44004f004d00410049004e00`	*Domain Name Data* ("DOMAIN")
76	`0x7500730065007200`	*User Name Data* ("user")
84	`0x57004f0052004b00530054004100540049004f004e00`	*Workstation Name Data* ("WORKSTATION")
106	`0xc337cd5cbd44fc9782a667af6d427c6de67c20c2d3e77c56`	*LM Response Data*
130	`0x25a98c1c31e81847466b29b2df4680f39958fb8c213a9cc6`	*NTLM Response Data*

Arti dari pesan Tipe III tersebut adalah:

1. Pesan tersebut adalah pesan Tipe III (dari NTLMSSP *signature* dan indikator Tipe III).
2. *Client* menginformasikan bahwa *string* dikodekan dalam Unicode (dari *flag* Negotiate Unicode).
3. *Client* mendukung otentikasi NTLM (dari *flag* Negotiate NTLM).
4. *Domain* milik *client* adalah “DOMAIN”.
5. *Username* milik *client* adalah “user”.

6. *Workstation* milik *client* adalah "WORKSTATION".
7. Respon LM *client* adalah `"0xc337cd5cbd44fc9782a667af6d427c6de67c20c2d3e77c56"`.
8. Respon NTLM *client* adalah `"0x25a98c1c31e81847466b29b2df4680f39958fb8c213a9cc6"`.
9. *Empty session key* telah dikirim.

3. Protokol NTLM Versi 2

NTLM versi 2 terdiri dari tiga algoritma respon (NTLMv2, LMv2, dan NTLM2 *session response*) dan skema penandaan dan penyegelan NTLM2 *session security*. NTLM2 *session security* dinegosiasikan melalui *flag* Negotiate NTLM2 Key. Otentikasi NTLMv2 dapat dijalankan dengan memodifikasi nilai *registry*. Pengaturan *registry* antara *client* dan *server* harus kompatibel agar otentikasi dapat berjalan dengan baik. Tingkat kompatibilitasnya adalah sebagai berikut:

Tingkat	Dikirim oleh *client*	Diterima oleh *server*
0	LM NTLM	LM NTLM LMv2 NTLMv2
1	LM NTLM	LM NTLM LMv2 NTLMv2
2	NTLM	LM NTLM LMv2 NTLMv2
3	LMv2 NTLMv2	LM NTLM LMv2 NTLMv2
4	LMv2 NTLMv2	NTLM LMv2 NTLMv2
5	LMv2 NTLMv2	LMv2 NTLMv2

Saat NTLM2 *session security* dinegosiasikan, NTLM2 *session response* dapat digunakan pada level 0, 1, dan 2 sebagai ganti dari LM dan NTLM yang lebih lemah. Hal ini dapat melindungi dari *dictionary attack* dengan pemberian respon terhadap *challenge* yang bervariasi dengan penambahan *client nonce* acak pada perhitungan.

4. Analisis Protokol Otentikasi NTLM

Protokol otentikasi NTLM dianggap sudah cukup aman untuk melakukan proses otentikasi. Beberapa keunggulan dari protokol ini adalah:

1. Sandi-lewat *user* tidak pernah ditransmisikan secara langsung.
2. Penghitungan nilai hash sandi-lewat tidak seperti pada LM dengan mengkonversikan ke huruf kapital terlebih dahulu tetapi tetap sesuai penulisan sandi-lewat asli sehingga sulit untuk ditembus.
3. Protokol ini sederhana dan cukup aman jika sandi-lewat yang dipakai oleh *user* cukup kuat.

Akan tetapi, protokol ini juga masih memiliki kekurangan-kekurangan. Kekurangan protokol NTLM antara lain:

1. Otentikasi harus dilakukan untuk setiap penggunaan *server*, hal ini dapat menyebabkan *bottleneck* pada *domain controller*.
2. Sandi-lewat yang lemah akan mudah diserang menggunakan *dictionary attack* secara *offline*.
3. Protokol ini tidak bisa digunakan untuk melakukan otentikasi *server*.

5. Serangan Kriptografi Terhadap Protokol NTLM

Protokol NTLM walaupun lebih aman dari protokol LM masih tidak luput dari berbagai serangan kriptografi yang digunakan untuk menembus proses otentikasi. Serangan yang lazim digunakan untuk menembus protokol ini diantaranya adalah *dictionary attack* dan *middle person attack.*

6. Kesimpulan

1. Protokol otentikasi berperan penting dalam komunikasi jaringan untuk menjaga keamanan pertukaran data.
2. Protokol NTLM lebih aman daripada protokol LM karena dalam penghitungan hash sandi-lewat tidak dikonversi ke dalam huruf kapital terlebih dulu sehingga lebih sulit diserang.
3. Protokol NTLMv2 lebih aman daripada NTLM karena menggunakan blok data yang disebut 'blob'.
4. Komunikasi dalam jaringan lebih aman jika mengimplementasikan protokol otentikasi NTLMv2 karena sukar ditembus.

6. Daftar Pustaka

[1] Munir, Rinaldi. Diktat Kuliah IF5054 Kriptografi. Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. 2006

[2] http://davenport.sourceforge.net/ntlm.html

[3] http://en.wikipedia.org/wiki/NTLM

[4] http://thesource.ofallevil.com/technet/ prodtechnol/windows2000serv/reskit/ prork/prdd_sec_jxbq.mspx

[5] http://www.opengroup.org/comsource/ techref2/ NCH1222X.HTM