We start by defining a variable <text> to hold the ciphertext, <chars> to hold those characters we're interested in (letters and the underscore), and <threes> as the characters broken up into their letter-parts - so A becomes A1, A2, A3, implemented as {"A"[1], "A"[2], "A"[3]}.

In[28]:=
```
text =
  "WJX SUE IXNSON DHZDUFZ_L KU YTL VLGWSBBAT XWD JML UWFRO_CHQVH ODZRGKT – EUU
    BDTQXPS ELV MDSGB NDVA IOLQY_NRZ EUXXSOOX TDWKCZTL MU TH. WGW_ AJX
    NZYVT YMWSQTX. G ULPWO QI PFSUVK USNPYFPO GO VTIWX DDFVB _SWCLXO_Q
    FT PUW SAAVTB FA EQK _AB QXO, WKN WW FYWLES PZGOSOUN RQ WB NDVR
    JLZEGQWLP QXD S_IOXMC XUD MMLB NZY X_B MQ ENK PLDUWME. D SLAJWVD
    AQF WLKP VCLAS QI PFSUUK XZLTXUPO UO VTSUK DTZUCLYPV, SFC WGX_ XVUX
    IMD_AAZ O_ DCEYWNMKX OTK M ARB LPVEK WDVF NZ RIILLJI_H VTSWK UUDSN
    ZPOYTD RZWBJMMXAR ZENST. TUV OWDBHWFRY JKKALW NAM ZEAOUBKX AQ WGX
    REAJSKLVGMPLA ITO AH NTSZYG NUDUC BFEFIXW AIUILEH NB. YTYU IDEE
    UTCAWW HBA RBCH I NDXS, BKO W DBE GTCAWU IDVW GWIM WGX_ HIOXF RE
    OB RJWG. IHIVB NZYY YFEGO KFP FY UXTSTZT V_JX GMOLQ W KOU_N SVVR
    AAAUSD XNWG L YTV_D OBJUU ITYW DAX NON DAWV NVOZ UCKQOQ LSUO_BS QO.
VFY _EOXLHJ UHKE XUVG LQWIA PWJFGJ, THNX IKOALCCOX TS WGX RQF YXZHID FU
    QENXWOWBTX. Q AEB OBQVXT RXUN ITYV HAD VMCAXLW NF RFZX CSTJFD WXTWMPF
    IX JOS L IAS QKKZCI LF OFUB MA KQH DSZWJ WR JMUXF SV NBDUWC NZYJ. DHQS
    VBOX TEO VEGUOKWDVF, LAW OG W DVVE HAD QG GZSACKQ ZVQ UGA QJ DAHPXGOZA
    VB L_DB. W DVVE NVOOU BETJP NZ RIILLJISH W OEV WLRGNWKT QZD P OEW AQXL
    UFK SEQ, LRW UCLEOIV HAS FXA VTV UXMMUHCME OYY XNDB TZEE. LHQS OYY AM
    THNF Y_WB ZPJ ENDST IB BC QJ UUA, HMN PIEQWGJ OYY Z_DD W ANBED DCUT
    UUA QJ JXMB OU OTVSTAR O_ NEY O_NELSBPWLUC BDESHCAE. WGXY HRAX ZPLV
    FB TOWGZE SBRX EEQ LRW WE MXS XUVQ EUU FB_VONJGI_U MQ QRKF YSNXYTVV.
W OO MZKKOI LHQS V SLEH XUD MQ UPFRRZ U TNZ XNMQ ZVQ UGA O_ FRDVW NN
    RWVGDSI UHM ZYQYESGNB MKH RWLL UOXX, M NUGX EEQ DPM÷F JWMW. V SLEH
    AGRJL WJX _RQCKUF_NNF HV QFRJUH NLY NCYDDUS FF TUW UUA_O MPO, XUKKS,
    WHFER ZUXCUTJFX J TRNZPIIXA AURWGJ WIX SBW. VX SBO GF RGAEKO AKN
    WA TVUMF AITNJKK _FI SIS TDOI WF KLVR FB. WISH MQKNXFR WVPDSIM,

NWXVVWMC.
";
(*chars holds the characters which are encrypted – that is, letters and _ *)
chars = Select[Characters[text], LetterQ[#] || # === "_" &];
(*threes holds those encrypted characters but broken up into their letter-
  parts: "A" → {"A"[1],"A"[2],"A"[3]}*)
threes = Flatten[{#[1], #[2], #[3]} & /@ chars];
(*transpositionStage is a function which carries out
 the transposition: reversing the five-tuples of the text*)
```

We also define a function transpositionStage which performs the given transformation specified in <order> on the given <listOfChars>, with a default interpretation of reversing the list - that is, <order> is {5, 4, 3, 2, 1}. It works by Partitioning the list into sublists of appropriate length, applying the transformation to them one by one, and then squidging the lists back together again using Flatten.

In[31]:=
```
transpositionStage[listOfChars_List] :=
 transpositionStage[listOfChars, {5, 4, 3, 2, 1}]
transpositionStage[listOfChars_List, order_List] :=
 Flatten[Extract[#, List /@ order] & /@ Partition[listOfChars, Length[order]]]
```

Finally, we define a variable <untransposed> as the ciphertext with the transposition stage removed, and Partitioned into groups of three.

In[33]:=
```
(*untransposed holds the letter-
 parts of the cipher with the transposition stage removed*)
untransposed = Partition[transpositionStage[threes], 3];
```

This is what the start and end of the list <untransposed> looks like:

In[34]:=
```
untransposed // Short
```

Out[34]//Short=
```
{{J[2], J[1], W[3]}, {W[2], W[1], S[1]}, {X[3], X[2], X[1]},
  <<1405>>, {V[3], C[3], C[2]}, {C[1], M[3], M[2]}}
```

Now we begin the cracking process. We formalise our cribs: “TIBERIUS” appears in positions -8 through -1, and “VALENTINEANDPROTEUS” in positions 222 through 240. We represent the cribs as lists of replacement rules:

In[35]:=
```
(*we have a crib: the word "TIBERIUS" being the last 8 characters;
we split that up into its letter-parts so that the crib
  "IUS becomes WMC" is expressed as "C[1] is decrypted to S[1]" etc
  Note that we've already de-applied the transposition,
so we're left with a straight substitution
*)
firstCrib = {Thread /@ Thread[untransposed[[-8 ;;]] →
      (Function[{ch}, ch[#] & /@ Range[3]] /@ Characters["TIBERIUS"])]};
(*do the same with VALENTINEANDPROTEUS - this is in position 222 to 240*)
secondCrib = {Thread /@
    Thread[untransposed[[222 ;; 240]] → (Function[{ch}, ch[#] & /@ Range[3]] /@
        Characters["VALENTINEANDPROTEUS"])]};
```

So firstCrib looks like this:

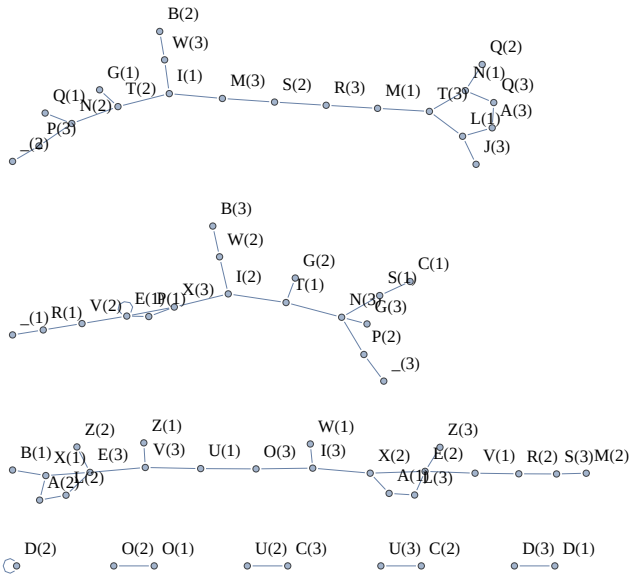
In[38]:=
```
firstCrib
```

Out[38]=
```
{{{N[3] → T[1], N[2] → T[2], N[1] → T[3]}, {M[3] → I[1], X[3] → I[2], X[2] → I[3]},
  {X[1] → B[1], W[3] → B[2], W[2] → B[3]}, {V[2] → E[1], V[1] → E[2], V[3] → E[3]},
  {V[2] → R[1], V[1] → R[2], M[1] → R[3]}, {W[3] → I[1], W[2] → I[2], W[1] → I[3]},
  {V[3] → U[1], C[3] → U[2], C[2] → U[3]}, {C[1] → S[1], M[3] → S[2], M[2] → S[3]}}}
```

Now we formulate a graph, where the vertices are the letter-parts like "M"[3]. We define two vertices (letter-parts), x and y, to be connected iff the crib has given us a replacement rule from one to the other: x → y. Side note: we have essentially defined an equivalence relation: “letter-parts x and y are related iff their coordinate in the keycube is the same” - so, for example, if A were in position {1, 1, 2} and B were in position {3, 2, 1} in the cube, then “A”[1] is related to “B”[3] and “A”[2]; we later use the fact that equivalence classes partition the set, but we use a Graph structure to represent the equivalence classes.

In[39]:=
```
(*the cribs have given us a list of letter-
 parts which are the same as each other - for example,
M[3]→S[2]. A graph is a convenient way to express this;
we need to turn the Rules that it is currently expressed as,
into UndirectedEdges;
we also remove duplicates because we are sometimes told the same thing twice
 (eg. A[1]→N[2] appearing twice in the list - I made up that example)
*)
graph = With[{rules = Flatten[{firstCrib, secondCrib}]},
  Graph[Sort /@ UndirectedEdge @@@ rules // DeleteDuplicates,
   VertexLabels → "Name", ImagePadding → 20]]
```

Out[39]=



Note that some nodes are on their own (for example, "D"[3]) - this is because our cribs haven't quite given us enough information to solve the cipher completely.

Now, we go through <untransposed>, which is the text with the transposition removed and divided up into chunks of three. For each chunk of three <untransposedTriplet>:
For each of the three letter-parts in <untransposedTriplet>, we use VertexComponent to extract from the graph all those nodes to which the letter-part is indirectly or directly connected (ie. the equivalence class of that letter-part). From those extracted nodes, we select those which could appear in the nth position.
So, for example, say we take in <untransposedTriplet> = {"A"[1], "C"[1], "B"[3]}. Looking at "A"[1], we find its VertexComponent (all those nodes to which "A"[1] is connected indirectly or directly). From that VertexComponent, we find all the letter-parts that could fit in the first slot - that is, "W"[1] is valid because it can appear in the first slot, but "Z"[3] is not because it must go in the third slot. Once we've done this for all three of the letter-parts, we may have ended up with something like <possibles> = {{"A"[1], "D"[1]}, {"A"[2]}, {"A"[3], "D"[3], "Z"[3]}}. We select the letter (there can be a maximum of one such letter) which appears in all three of the lists (in this case, "A"), because that is the only letter which will do as the letter that <untransposedTriplet> represents.

In[40]:=
```
filtered = Function[{untransposedTriplet}, With[{possibles =
      (*for each component of the triplet,
      using VertexComponent we select all the vertices which are connected to
        that component - this means possibles will contain every letter-part
        which can go in each of the three slots in the untransposedTriplet.
         Then we use the Select statement to pick out the letter-
       parts which can go in each slot. (For example, if we have
           {A[2],M[1],L[3]}, we only want anything that ends in a 1 to go
            in the first position - no letter consists of {A[2],A[2],A[3]})
      *)(Function[{whichSlot}, Select[VertexComponent[graph,
           untransposedTriplet[[whichSlot]]],
          First[#] == whichSlot &]] /@ Range[3])},

    (*Select from the possible substitutes for the first letter-
     part those elements which have a counterpart in the second letter-
     part and the third letter-part - so, for example,
    {{A[1]},{A[2],B[2]},{A[3],C[3]}} would have A[1] selected*)
    Select[possibles[[1]], MemberQ[Head /@ possibles[[2]], Head[#]] &&
       MemberQ[Head /@ possibles[[3]], Head[#]] &]]
  (*and do this for every triplet*) /@ untransposed
```

Out[40]=

A very large output was generated. Here is a sample of it:

```
{VertexComponent[], {}, {E[1]}, {}, {A[1]}, {S[1]}, {W[1]}, {E[1]},
 {A[1]}, {P[1]}, {}, {}, {D[1]}, {E[1]}, {}, {}, {}, VertexComponent[],
 {}, {E[1]}, VertexComponent[], {}, {}, VertexComponent[], {}, {E[1]},
 {}, {A[1]}, {B[1]}, {}, <<1350>>, {V[1]}, {E[1]}, VertexComponent[],
 {N[1]}, {W[1]}, {I[1]}, {T[1]}, VertexComponent[], {}, {}, {},
VertexComponent[], {E[1]}, {S[1]}, {}, {R[1]}, {}, {G[1]}, {A[1]}, {}, {},
 {S[1]}, {T[1]}, {I[1]}, {B[1]}, {E[1]}, {R[1]}, {I[1]}, {U[1]}, {S[1]}}
```

Show Less | Show More | Show Full Output | Set Size Limit...

Just to make it look a bit nicer:

In[41]:=
```
(*Pick out all the elements which have been definitively found – that is,
those which consist of {A[1]} where A is some letter.*)
If[Head[#] === List && Length[#] == 1, Head[#[[1]]], "_"] & /@ filtered // StringJoin //
 ToLowerCase
```

Out[41]=
```
__e_asweap__de_____e_____e_ab_lists__d____i__i__t_____t__et__s__angere_ma___t
  __ta__l__te__ve__o_e___e__e_b___t_iswa_t__ir_n__ingi__g__t_ga____ac___n_so_t
  _e_e____sap_ea__ng_nt__b__i_s_____wn_en___i____amea_pa___t__m_t
  ___valentineandproteus_ad__l_t__gast____en__i_s__a__red____an__ea_st_ga___
  __vi_enc_o_________a___r_bu_t_e_w_re____le_i_de___t____n__wasneve____et_esta
  __is_t_ei_____tbe__n__e_s__able___btt_______i_______e_t__v_l___le__t_ees__bl
  _s_ment_n____nea__t____s____e_b_tw_e__st___w____n_ure__wm_____ne_a__iw_s_nsu
  __w__tris_t_e__o_e__o_s__t_w__let_____l_______e_tainw_a_p____i____t_avel___
  _dwit_at_i_dp________di___t__ret_________a_ai_stme__w_owe________av_gr_wns_r
  ____it_i__lue__eatt_et___e___s____ve__mentiw_sw_rne_t_at________e_i_e_t___
  _ea____n_gains__e___iwas____e_______m___nd_a__i____e____sc__et_____tle_t_o
  _v_l_____ea__s_i__ve___t_pre_a____r_____d__a_pe_raswe__i__vet____s_ep_t
  _esta__is___ewi_en_it_and__ewli________an___a__es_as__lt__re_____s_o__i
  __nee_w_at_o_____t___isnew_ta_tisup_______t_n_wi___o__e__i____du_________i
  _____ien_si_t__i_telligen__s___i__st_e__avenee___peopleli_e___an_wema__av_t
  ____po_t__it_t_w_____et_e_i__a_____t_ati_a___ad_____s_an_wname__r___i___
  _ert_pr_pa__t_____ments____ill__e_i__p____d_n__in_i_a__t__e___e_p_____n_t
  _________t_____________gm__________e_xe_u_i___witnesse____i__t_ewar
  _ewas____w_____its__m__i_t_____r_i_n__e__l_ve_nwit_____es_r_ga__stiberius
```

This, with the word spacings, is enough to tell us that the first letters are THEGASWEAPONDE; we use this as another crib.

In[42]:=
```
thirdCrib = {Thread /@ Thread[untransposed[[1 ;; 14]] →
       (Function[{ch}, ch[#] & /@ Range[3]] /@ Characters["THEGASWEAPONDE"])]};

graph = With[{rules = Flatten[{firstCrib, secondCrib, thirdCrib}]},
   Graph[Sort /@ UndirectedEdge @@@ rules // DeleteDuplicates,
    VertexLabels → "Name", ImagePadding → 20]];

filtered = Function[{untransposedTriplet}, With[{possibles =
       (*for each component of the triplet, using VertexComponent we select
         all the vertices which are connected to that component –
        this means possibles will contain every letter-part which
         can go in each of the three slots in the untransposedTriplet.
          Then we use the Select statement to pick out the letter-
        parts which can go in each slot. (For example, if we have
            {A[2],M[1],L[3]}, we only want anything that ends in a 1 to go
             in the first position – no letter consists of {A[2],A[2],A[3]})
       *)(Function[{whichSlot}, Select[VertexComponent[graph,
            untransposedTriplet[[whichSlot]]],
           First[#] == whichSlot &]] /@ Range[3])},

     (*Select from the possible substitutes for the first letter-
      part those elements which have a counterpart in the second letter-
      part and the third letter-part – so, for example,
     {{A[1]},{A[2],B[2]},{A[3],C[3]}} would have A[1] selected*)
     Select[possibles[[1]], MemberQ[Head /@ possibles[[2]], Head[#]] &&
        MemberQ[Head /@ possibles[[3]], Head[#]] &]]
    (*and do this for every triplet*) /@ untransposed;
If[Head[#] === List && Length[#] == 1, Head[#[[1]]], "_"] & /@ filtered // StringJoin //
 ToLowerCase
```

Out[45]=
```
thegasweapondev___pe_____e_abulistshadone_is_inctiv__e_tu_eth_st_angere_ma___th
  _ta__l__te_ever_one___ec_e_b_itthiswast__ir_n_oingibegant_ga_he_acc__n_so
  _the_e____sappearing_nthebo_i_so_ou_ownmena__i____ameapparenttom_th
  _tvalentineandproteushadsol_t__gastoou_en__ies__abored_o__an__earst_ga_he_evi
  _enceo_the__t_eac__r_bu_the_weres___le_indec__t__n_n__wasneve____et
  _establishthei_gu__tbe__n__easonable_oubtthe___u_is____o_e_toovalua_le
  _otheest_bl_shmentan_anunea__t__c_se___e_between_st___we_eunsurehowmuchi_ne
  _a__iw_sunsur_wh_tris_the_po_e__ousbothwhilet_____l_n_____ertainwhatproo_im
  _ghthavelo_gedwit_at_irdpart_th__di_n_t__reta_e_ct_onagainstme__w_owever_he
  _havegrownstr_ngwithi__luenceattheto__e_e_s__governmentiwaswarne_thatth__h
  ___eci_e_to___eact__n_gains__ean_iwas_o__e_t____em__wndeathinor_er_o_sc_pet
  ___th_tle_t_ouvulne____ea__s_ih_veh__toprepa_e_or_outodisappearaswe__ih_veta
  _enstepstoestablishanewi_en_it_andanewli_e__r_ouan_charleshas__ltheresources
  _o__i__nee_what_o__ow_t__hisnew_ta_tisupto_oubut_nowing_o__e__i_ou_du_ge
  _outojoinm__rien_sint__intelligenc_se_vicest_e__avenee___peopleli_e_ouan
  _wema_havethe_pportunit_tow____o_et_eriama__a__thatihavehadtoch_oseanewname
  _or_ouin___ertopr_pa_ethe____ments__uwillnee_ihope_oudon__in_ihaveta
  _entheoppo___n_t__oh_nourt__c____g____he_oungmanh__r_wh_seexecuti_niwitnesse
  __uringthewarhewasn_cowa__a__itse_ms_itt_____rhi_na_e__l_ve_nwith_o__es_r_ga
  __stiberius
```

And now it's clearly "THEGASWEAPONDEVELOPEDBYTHEFABULISTS":

In[46]:=
```
thirdCrib =
  {Thread /@ Thread[untransposed[[1 ;; 35]] → (Function[{ch}, ch[#] & /@ Range[3]] /@
        Characters["THEGASWEAPONDEVELOPEDBYTHEFABULISTS"])]};

graph = With[{rules = Flatten[{firstCrib, secondCrib, thirdCrib}]},
   Graph[Sort /@ UndirectedEdge @@@ rules // DeleteDuplicates,
    VertexLabels → "Name", ImagePadding → 20]];

filtered = Function[{untransposedTriplet}, With[{possibles =
       (*for each component of the triplet, using VertexComponent we select
         all the vertices which are connected to that component –
        this means possibles will contain every letter-part which
         can go in each of the three slots in the untransposedTriplet.
          Then we use the Select statement to pick out the letter-
        parts which can go in each slot. (For example, if we have
            {A[2],M[1],L[3]}, we only want anything that ends in a 1 to go
             in the first position – no letter consists of {A[2],A[2],A[3]})
       *)(Function[{whichSlot}, Select[VertexComponent[graph,
            untransposedTriplet[[whichSlot]]],
           First[#] == whichSlot &]] /@ Range[3])},

     (*Select from the possible substitutes for the first letter-
      part those elements which have a counterpart in the second letter-
      part and the third letter-part – so, for example,
     {{A[1]},{A[2],B[2]},{A[3],C[3]}} would have A[1] selected*)
     Select[possibles[[1]], MemberQ[Head /@ possibles[[2]], Head[#]] &&
        MemberQ[Head /@ possibles[[3]], Head[#]] &]]
   (*and do this for every triplet*) /@ untransposed;

If[Head[#] === List && Length[#] == 1, Head[#[[1]]], "_"] & /@ filtered // StringJoin
```

Out[49]=
```
THEGASWEAPONDEVELOPEDBYTHEFABULISTSHADONEDISTINCTIVEFEATURETHESTRANGEREDMARKST:
HATAFFLICTEDEVERYONEAFFECTEDBYITTHISWASTHEIRUNDOINGIBEGANTOGATHERACCOUNTSOF:
THESEMARKSAPPEARINGONTHEBODIESOFOUROWNMENANDITBECAMEAPPARENTTOMETHATVALENTI:
NEANDPROTEUSHADSOLDTHEGASTOOURENEMIESILABOREDFORMANYYEARSTOGATHEREVIDENCEOF:
THEIRTREACHERYBUTTHEYWERESKILLEDINDECEPTIONANDIWASNEVERABLETOESTABLISHTHEIR:
GUILTBEYONDREASONABLEDOUBTTHEFABULISTSPROVEDTOOVALUABLETOTHEESTABLISHMENTAN:
DANUNEASYTRUCESETTLEDBETWEENUSTHEYWEREUNSUREHOWMUCHIKNEWANDIWASUNSUREWHATRI:
SKTHEYPOSEDTOUSBOTHWHILETHEYCOULDNOTBECERTAINWHATPROOFIMIGHTHAVELODGEDWITHA:
THIRDPARTYTHEYDIDNOTDARETAKEACTIONAGAINSTMENOWHOWEVERTHEYHAVEGROWNSTRONGWIT:
HINFLUENCEATTHETOPLEVELSOFGOVERNMENTIWASWARNEDTHATTHEYHADDECIDEDTOTAKEACTIO:
NAGAINSTMEANDIWASFORCEDTOFAKEMYOWNDEATHINORDERTOESCAPETHEMTHATLEFTYOUVULNER:
ABLEANDSOIHAVEHADTOPREPAREFORYOUTODISAPPEARASWELLIHAVETAKENSTEPSTOESTABLISH:
ANEWIDENTITYANDANEWLIFEFORYOUANDCHARLESHASALLTHERESOURCESYOUWILLNEEDWHATYOU:
DOWITHTHISNEWSTARTISUPTOYOUBUTKNOWINGYOUWELLIWOULDURGEYOUTOJOINMYFRIENDSINT:
HEINTELLIGENCESERVICESTHEYHAVENEEDOFPEOPLELIKEYOUANDWEMAYHAVETHEOPPORTUNITY:
TOWORKTOGETHERIAMAFRAIDTHATIHAVEHADTOCHOOSEANEWNAMEFORYOUINORDERTOPREPARETH:
EDOCUMENTSYOUWILLNEEDIHOPEYOUDONTMINDIHAVETAKENTHEOPPORTUNITYTOHONOURTHECOU:
RAGEOFTHEYOUNGMANHARRYWHOSEEXECUTIONIWITNESSEDDURINGTHEWARHEWASNOCOWARDANDI:
TSEEMSFITTINGFORHISNAMETOLIVEONWITHFONDESTREGARDSTIBERIUS
```

Incidentally, we can deduce that the transposition is {1,2,3,4,5}->{5,4,3,2,1} because if we perform this process with an incorrect transposition, the resulting string does not contain the string “valentineandproteus” - the letter-parts aren’t in the right places to allow this. It turns out that {5,4,3,2,1} is the only permutation which results in “valentineandproteus” appearing somewhere in the text (when the two cribs are applied) for all permutations of length up to 6. mad_maths_man gives some good reasoning based on the Kasiski test for why the transposition keylength should be 5.

We can deduce that the ciphertext is read off in rows and not columns because of the string “NZ

RIILLJI_H", which is repeated (or very nearly) in two different places in the ciphertext. If the ciphertext were read off in columns, the repeated characters would have been flung far and wide over the cipher. In this way, given knowledge that the cipher is ADFGVX-style but with a 3x3x3 grid, we can crack the cipher.

A function to decrypt the whole thing:

In[52]:=
```
decrypt8B[txt_] :=
 decrypt8B[txt, Graph[(*this is the graph that holds every dependency,
  but zoomed out really far*)]]
decrypt8B[txt_, graph_] :=
 With[{chars = Select[Characters[ToUpperCase@txt], LetterQ[#] || # === "_" &]},
  With[{threes = Flatten[{#[1], #[2], #[3]} & /@ chars]},
   With[{untransposed = Partition[transpositionStage[threes], 3]},

    With[{filtered = Function[{untransposedTriplet}, With[{possibles =
          (*for each component of the triplet, using VertexComponent we
            select all the vertices which are connected to that component -
           this means possibles will contain every letter-part which
            can go in each of the three slots in the untransposedTriplet.
             Then we use the Select statement to pick out the letter-parts
            which can go in each slot. (For example, if we have {A[2],M[1],
                L[3]}, we only want anything that ends in a 1 to go in the
                first position - no letter consists of {A[2],A[2],A[3]})
          *)(Function[{whichSlot}, Select[VertexComponent[
               graph, untransposedTriplet[[whichSlot]]],
             First[#] == whichSlot &]] /@ Range[3])},

        (*Select from the possible substitutes for the first letter-
         part those elements which have a counterpart in the second letter-
         part and the third letter-part - so, for example,
        {{A[1]},{A[2],B[2]},{A[3],C[3]}} would have A[1] selected*)
        Select[possibles[[1]], MemberQ[Head /@ possibles[[2]], Head[#]] &&
           MemberQ[Head /@ possibles[[3]], Head[#]] &]] /@ untransposed},
     If[Head[#] === List && Length[#] == 1, Head[#[[1]]], "_"] & /@ filtered //
       StringJoin // ToLowerCase]]]]
```

In[54]:= **decrypt8B[text]**

Out[54]= thegasweapondevelopedbythefabulistshadonedistinctivefeaturethestrangeredmarkst
hatafflictedeveryoneaffectedbyitthiswastheirundoingibegantogatheraccountsof
thesemarksappearingonthebodiesofourownmenanditbecameapparenttomethatvalenti
neandproteushadsoldthegastoourenemiesilaboredformanyyearstogatherevidenceof
theirtreacherybuttheywereskilledindeceptionandiwasneverabletoestablishtheir
guiltbeyondreasonabledoubtthefabulistsprovedtoovaluabletotheestablishmentan
danuneasytrucesettledbetweenustheywereunsurehowmuchiknewandiwasunsurewhatri
sktheyposedtousbothwhiletheycouldnotbecertainwhatproofimighthavelodgedwitha
thirdpartytheydidnotdaretakeactionagainstmenowhowevertheyhavegrownstrongwit
hinfluenceatthetoplevelsofgovernmentiwaswarnedthattheyhaddecidedtotakeactio
nagainstmeandiwasforcedtofakemyowndeathinordertoescapethemthatleftyouvulner
ableandsoihavehadtoprepareforyoutodisappearaswellihavetakenstepstoestablish
anewidentityandanewlifeforyouandcharleshasalltheresourcesyouwillneedwhatyou
dowiththisnewstartisuptoyoubutknowingyouwelliwouldurgeyoutojoinmyfriendsint
heintelligenceservicestheyhaveneedofpeoplelikeyouandwemayhavetheopportunity
toworktogetheriamafraidthatihavehadtochooseanewnameforyouinordertoprepareth
edocumentsyouwillneedihopeyoudontmindihavetakentheopportunitytohonourthecou
rageoftheyoungmanharrywhoseexecutioniwitnessedduringthewarhewasnocowardandi
tseemsfittingforhisnametoliveonwithfondestregardstiberius